

# Raduno OpenDataSicilia 2022

Dati covid e controlli di qualità  
**FRICTIONLESS**

**ALDO MARIA BRACCO**

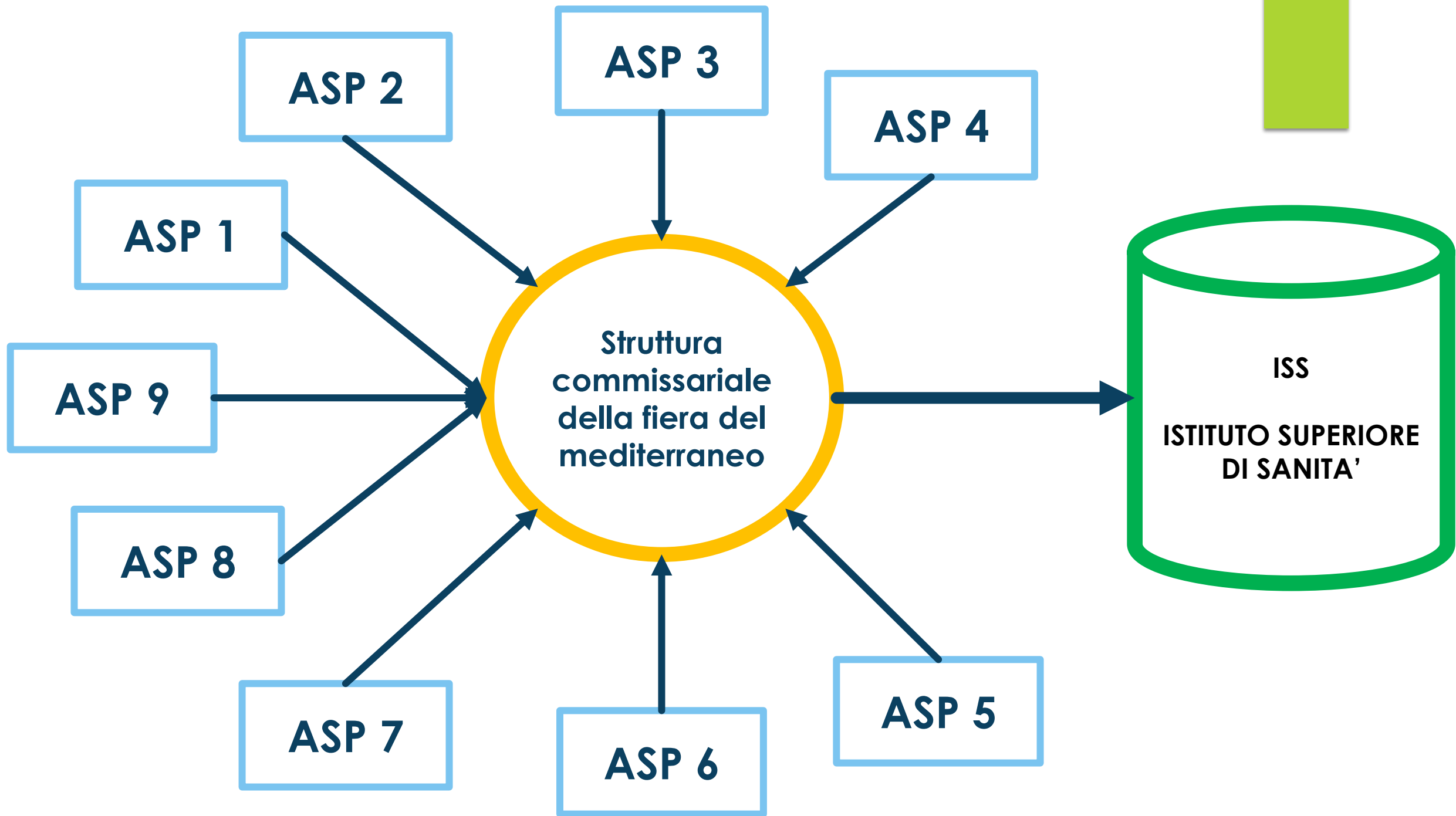
CONTATTI:

-EMAIL : **ALDOMARIA.BRACCO@GMAIL.COM**

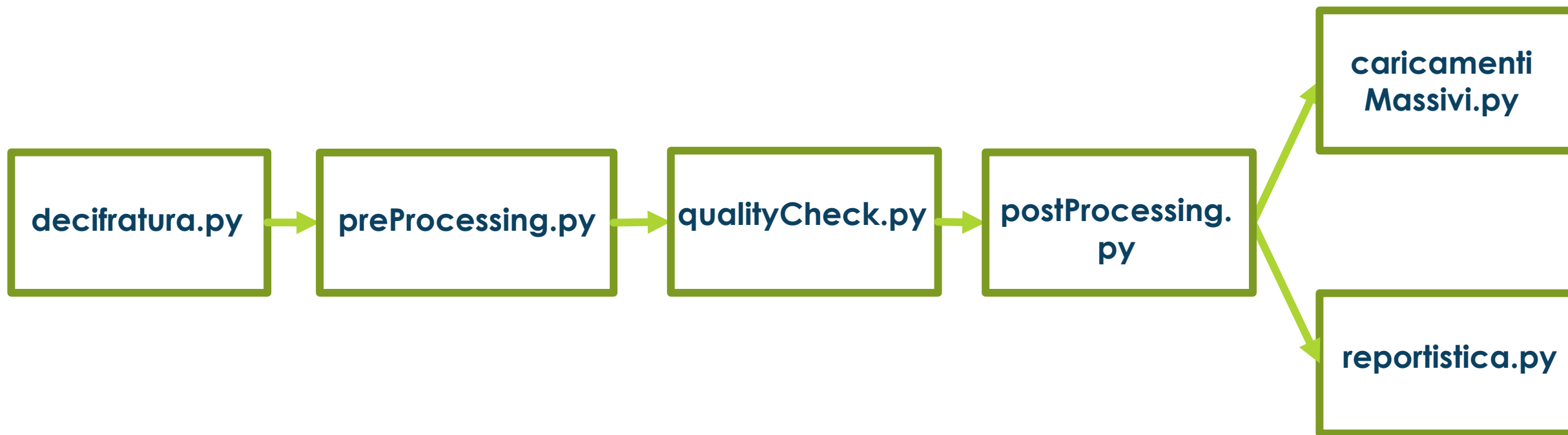
- PEC : **ALDOMARIA.BRACCO@ORDINEINGPA.IT**

# CONTESTO APPLICATIVO

- ▶ Pandemia Covid-19
- ▶ Struttura Commissariale all'Emergenza Covid-19
- ▶ Gestione pazienti con infezione respiratoria da SARS-CoV-2:
  - ▶ Cure mediche, tracciamento, supporto psicologico, isolamento, convocazioni, etc...
- ▶ Gestione del sistema informativo
- ▶ Monitoraggio statistico dello stato pandemico
- ▶ Caricamenti massivi per l'intera regione Sicilia sul portale dell'Istituto Superiore di Sanità



# PIPELINE DI ELABORAZIONE



# FRICTIONLESS

- ▶ Framework **data management** per Python
- ▶ Open source
- ▶ Riduzione attrito durante le fasi di lavoro
- ▶ Permette un miglior **utilizzo** dei dati e la loro corretta **pubblicazione** o **condivisione**
- ▶ Miglioramento dell'interoperabilità di dati e **metadati**
- ▶ Frictionless standard

# FRICTIONLESS

- ▶ Versione corrente **v4.40.8**
  - ▶ `pip install frictionless`
- ▶ Versione beta **v5**
  - ▶ `pip install frictionless --pre`
- ▶ Frictionless può essere usato come:
  - ▶ Libreria Python
  - ▶ interfaccia a riga di comando
  - ▶ API
- ▶ <https://github.com/frictionlessdata/framework>

# FRICTIONLESS STANDARD

- ▶ Questi standard sono un insieme di modelli per la descrizione dei dati
- ▶ **TABLE SCHEMA** : un semplice formato per dichiarare un schema per dati tabulari
- ▶ **DATA RESOURCE** : un semplice formato per descrivere e impacchettare una singola risorsa
- ▶ **DATA PACKAGE** : un formato contenitore per descrivere una raccolta coerente di dati

# FRICTIONLESS FRAMEWORK

- ▶ Il framework garantisce qualità dei dati tramite operazioni di trasformazione e/o validazione e la generazione di metadati ad essi associati
- ▶ **4 funzioni** principali (**DEVT**) da utilizzare anche indipendentemente:
  - ▶ **DESCRIBE**
  - ▶ **EXTRACT**
  - ▶ **VALIDATE**
  - ▶ **TRASFORM**



# ESEMPIO

## ► data.csv

	A	B	C	D
1	id	esito_controllo	data_controllo	residenza
2	23232	0	05/09/2021	PA
3	62865	1	11/02/2022	RM
4	X_d3321	0	11/05/2022	ME
5	85356	0	11/05/2022	PA
6	15752	0	13/02/2021	ME
7	23123	1	11/02/2022	CT
8	76363	1	14/02/2022	RG

id;esito\_controllo;data\_controllo;residenza  
23232;0;2021-09-05;PA  
62865;1;2022-02-11;RM  
...

...

## sicilia\_provinces.csv

	A	B
1	provincia	residenti
2	AG	412427
3	CL	250550
4	CT	1068835
5	EN	155982
6	ME	599990
7	PA	1199626
8	RG	315082
9	SG	383743
10	TP	415233

# DESCRIBE

- ▶ Descrivere i dati significa generare **metadati** associati ad essi
- ▶ I metadati forniscono le informazioni sufficienti per comprendere a pieno i dati
- ▶ È possibile specificare il **tipo** di dati, **vincoli** e **relazioni** fra i dati e altro ancora
- ▶ È possibile descrivere uno Schema, un Resource o un Package
- ▶ Metadati salvati nel formato per la serializzazione di dati **YAML**



```
from frictionless import describe
```

```
package = describe("*.csv", type="package")
```

```
package.title = "Dati utenti e province regione sicilia"
```

```
package.description = "Questi dati sono dati esempio per mostrare il processo di  
verifica dei dati"
```

```
package.get_resource("sicilia_province").schema.get_field("residenti").type =  
"integer"
```

```
package.get_resource("sicilia_province").schema.get_field("provincia").constrain  
ts["required"] = True
```

```
package.get_resource("data").schema.foreign_keys.append(  
    {"fields": ["residenza"], "reference": {"resource": "sicilia_province",  
"fields": ["residenza"]}}  
)
```

```
package.get_resource("data").name = "pazienti"
```


```
package.get_resource("pazienti").schema.get_field("id").type = "integer"
```

```
package.get_resource("pazienti").schema.get_field("data_controllo").type =  
"date"
```

```
package.get_resource("pazienti").schema.get_field("esito_controllo").constraints  
["maximum"] = 1
```

```
package.to_yaml("pazienti_sicilia.package.yaml")
```

```
resources:
- path: data.csv
  name: pazienti
  profile: tabular-data-resource
  scheme: file
  format: csv
  hashing: md5
  encoding: utf-8
  dialect:
    delimiter: ;
  schema:
    fields:
      - type: integer
        name: id
      - type: integer
        name: esito_controllo
        constraints:
          maximum: 1
      - type: date
        name: data_controllo
      - type: string
        name: residenza
  foreignKeys:
    - fields:
        - residenza
      reference:
        resource: sicilia_province
        fields:
          - residenza
- path: sicilia_province.csv
```



```
    fields:
      - residenza
- path: sicilia_province.csv
  name: sicilia_province
  profile: tabular-data-resource
  scheme: file
  format: csv
  hashing: md5
  encoding: utf-8
  dialect:
    delimiter: ;
  schema:
    fields:
      - type: string
        name: provincia
        constraints:
          required: true
      - type: integer
        name: residenti
  profile: data-package
  title: Dati utenti e province regione sicilia
  description: Questi dati sono dati esempio per mostrare il processo di verifica dei
    dati
```

# EXTRACT

- ▶ Estrarre dati equivale a leggere dati tabulati da una fonte
- ▶ Questa funzione legge, gestisce e memorizza i dati in memoria come **rows**
- ▶ È possibile **estrarre i dati in base ai loro metadati**. Questo consente di ottenere dati puliti con i quali è possibile lavorare
- ▶ È possibile estrarre un Resource o un Package

# EXTRACT



```
from frictionless import extract
from tabulate import tabulate

data = extract('pazienti_sicilia.package.yaml')
print(tabulate(data, headers="keys", tablefmt="rst"))
```

# EXTRACT

```
=====
data.csv
=====
```

```
{'residenza': 'PA', 'id': 23232, 'esito_controllo': 0, 'data_controllo': datetime.date(2021, 9, 5)}
{'residenza': 'RM', 'id': 62865, 'esito_controllo': 1, 'data_controllo': datetime.date(2022, 2, 11)}
{'residenza': 'ME', 'id': None, 'esito_controllo': 0, 'data_controllo': datetime.date(2022, 5, 11)}
{'residenza': 'PA', 'id': 85358, 'esito_controllo': 0, 'data_controllo': datetime.date(2022, 5, 11)}
{'residenza': 'ME', 'id': 15752, 'esito_controllo': 0, 'data_controllo': None}
{'residenza': 'CT', 'id': 23123, 'esito_controllo': 1, 'data_controllo': datetime.date(2022, 2, 11)}
{'residenza': 'RG', 'id': 76363, 'esito_controllo': 1, 'data_controllo': datetime.date(2022, 2, 14)}
```

```
=====
sicilia_provinces.csv
=====
```

```
{'provincia': 'AG', 'residenti': 412427}
{'provincia': 'CL', 'residenti': 250550}
{'provincia': 'CT', 'residenti': 1068835}
{'provincia': 'EN', 'residenti': 155982}
{'provincia': 'ME', 'residenti': 599990}
{'provincia': 'PA', 'residenti': 1199626}
{'provincia': 'RG', 'residenti': 315082}
{'provincia': 'SG', 'residenti': 383743}
{'provincia': 'TP', 'residenti': 415233}
=====
```

# VALIDATE

- ▶ IL processo di validazione consiste nell'identificazione di eventuali problemi nei dati
- ▶ Questo processo garantisce qualità dei dati e velocizza le operazioni di correzione
- ▶ È possibile validare uno Schema, un Resource, un Package, un inquiry (un oggetto che rappresenta una sequenza di task di validazione), un report o un oggetto errors
- ▶ Viene generato un **report descrittivo** del processo di validazione esguito



# VALIDATE



```
from frictionless import describe, validate

# package descriptor creato prima tramite describe()

# validate
report = validate("pazienti_sicilia.package.yaml")

### write report in report.pazienti_sicilia.txt
f = open("report." + 'pazienti_sicilia' + ".txt", "w")
f.write(report.to_summary())
```

# REPORT VALIDAZIONE

```
# -----  
# valid: sicilia_province.csv  
# -----
```

## Summary

Description	Size/Name/Count
File name	sicilia_province.csv
File size (bytes)	122
Total Time Taken (sec)	0.005

```
# -----  
# invalid: data.csv  
# -----
```

## Summary

Description	Size/Name/Count
File name	data.csv
File size (bytes)	208
Total Time Taken (sec)	0.012
Total Errors	3
ForeignKey Error (foreign-key-error)	1
Type Error (type-error)	2

## Errors

row	field	code	message
3		foreign-key-error	Row at position "3" violates the foreign key: for "residenza": values "RM" not found in the lookup table "sicilia_province" as "provincia"
4	1	type-error	Type error in the cell "X_d3321" in row "4" and field "id" at position "1": type is "integer/default"
6	3	type-error	Type error in the cell "13/02/2021" in row "6" and field "data_controllo" at position "3": type is "date/default"

# TRANSFORM

- ▶ In Frictionless trasformare i dati significa modificare sia dati che metadati
- ▶ Durante il processo di trasformazione il framework offre miglior controllo sui metadati e consente di lavorare su dati di grandi dimensioni
- ▶ Il core della funzione transform utilizza il package **PETL** per estrarre, trasformare e caricare tabelle di dati
- ▶ È possibile trasformare un Resource, un Package o una **Pipeline** (un modo dichiarativo di descrivere gli step di trasformazione)

# TRANSFORM STEPS

- ▶ Frictionless offre più di **40 built-in step** di trasformazione
- ▶ **Resource steps** trasformazioni su Package. È possibile aggiungere, rimuovere, trasformare o aggiornare una risorsa
- ▶ **Table steps** : trasformazioni su Resource. È possibile lavorare su tabelle di dati (Join, Aggregate, Pivot, etc...)
- ▶ **Field steps** : solo per le trasformazioni su Schema. È possibile lavorare sulle colonne (Add, Move, Filter, etc...)
- ▶ **Row steps** : trasformazioni su righe
- ▶ **Cell steps** : trasformazioni sulle celle (Convert, Fill, Format, etc...)

# TRANSFORM

```
from pprint import pprint
from frictionless import Package, Resource, transform, steps

source = Resource(path="data.csv")
target = transform(
    source,
    steps=[
        steps.field_move(name="residenza", position=2),
        steps.table_normalize(),
        steps.row_filter(formula="esito_controllo > 0"),
    ]
)
print(target.schema)
print(target.to_view())
```

# TRANSFORM

target.schema

```
{'fields': [{'name': 'id', 'type': 'string'},  
            {'name': 'residenza', 'type': 'string'},  
            {'name': 'esito_controllo', 'type': 'integer'},  
            {'name': 'data_controllo', 'type': 'string'}]}
```

target.to\_view()

id	residenza	esito_controllo	data_controllo
'62865'	'RM'	1	'2022-02-11'
'23123'	'CT'	1	'2022-02-11'
'76363'	'RG'	1	'2022-02-14'

# PERCHÈ FRICTIONLESS

- ▶ Semplice da imparare
- ▶ Permette uno sviluppo in tempi rapidi
- ▶ Performante
- ▶ Completo delle funzionalità base
- ▶ Open source



**GRAZIE PER L'ATTENZIONE**