

Chapitre 2 :

Traitement des donnees

A. Notes de cours

- 1) les étapes impliquées dans le prétraitement des données sont : la gestion des valeurs manquantes, la fusion des ensembles de données, l'application de fonctions, les agrégations et le tri.
- 2) Pour importer les différentes fonctions et types de données requis à partir de `spark.sql` on utilise :
 - `from pyspark.sql import SparkSession`
 - `import pyspark.sql.functions as F`
 - `from pyspark.sql.types import *`
- 3) Dans Spark, nous pouvons traiter les valeurs nulles en les remplaçant par une valeur spécifique ou en supprimant les lignes/colonnes contenant des valeurs nulles.
 - remplacement avec : `replace()`
 - suppression avec : `drop()`
- 4) Un sous-ensemble d'une trame de données peut être créé, en fonction de plusieurs conditions dans lesquelles nous sélectionnons quelques lignes, colonnes ou données avec certains filtres en place.
- 5) le processus de filtrage des enregistrements :
 - Select : on utilise `select()`
 - Filter : on utilise `filter()`
 - Where : on utilise `where()`
- 6) Tout type d'agrégation peut être simplement divisé en trois étapes, dans l'ordre suivant :
 - Diviser
 - Appliquer
 - Combiner

- 7) Nous agrégeons également les données en utilisant les fonctions `groupBy()`
- 8) Il existe différents types d'opérations sur des groupes d'enregistrements
- Moyenne
 - Maximum
 - Mini
 - Somme
- 9) Nous pouvons collecter des valeurs d'une liste de deux manières différentes :
- Collect List : elle fournit toutes les valeurs dans l'ordre d'occurrence d'origine (elles peuvent également être inversées)
 - Collect Set : elle fournit que les valeurs uniques
- 10) User-Defined Functions (UDFs) : sont des routines programmables par l'utilisateur qui agissent sur une ligne. Pour définir les propriétés d'une fonction définie par l'utilisateur, l'utilisateur peut utiliser certaines des méthodes définies dans cette classe :
- `asNonNullable()` : met à jour `UserDefinedFunction` en non nullable.
 - `AsNondeterministic()` : met à jour `UserDefinedFunction` sur non déterministe.
 - `withName(name: String)`: met à jour `UserDefinedFunction` avec un nom donné.
- 11) Pandas UDFs sont beaucoup plus rapides et efficaces, en termes de temps de traitement et d'exécution, par rapport aux standard Python UDFs. La principale différence entre eux est que UDF Python est exécuté ligne par ligne et, par conséquent, n'offre vraiment pas l'avantage d'un framework distribué. Cela peut prendre plus de temps, par rapport à un Pandas UDF, qui s'exécute bloc par bloc et donne des résultats plus rapides. La seule différence dans l'utilisation réside dans la déclaration.
- 12) PySpark offre un moyen très pratique de fusionner et de faire pivoter vos valeurs de dataframe, selon les besoins.
- 13) Pivoting dans PySpark permet de créer simplement une vue pivot du cadre de données pour des colonnes spécifiques.

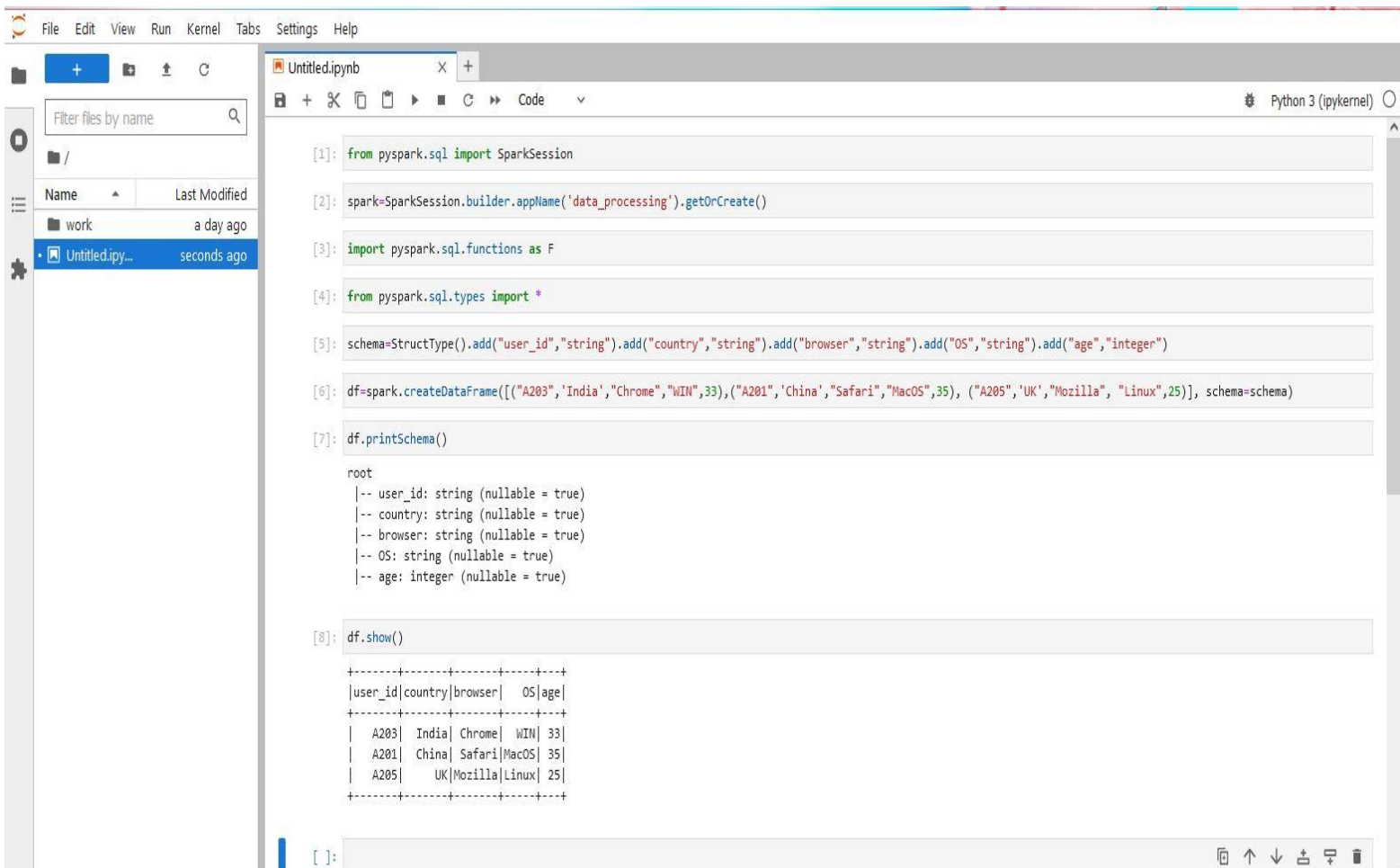
14) Window Functions or Windowed Aggregates : dans PySpark permet d'effectuer certaines opérations sur des groupes d'enregistrements appelés "dans la fenêtre". Il calcule les résultats pour chaque ligne dans la fenêtre.

15) PySpark prend en charge trois types de fonctions de fenêtre :

- Agrégations
- Classement
- Analytique

B. CAS PRATIQUE

➤ Création d'un objet sparksession et d'un cadre de donnée



The screenshot shows a Jupyter Notebook with the following code cells:

```
[1]: from pyspark.sql import SparkSession

[2]: spark=SparkSession.builder.appName('data_processing').getOrCreate()

[3]: import pyspark.sql.functions as F

[4]: from pyspark.sql.types import *

[5]: schema=StructType().add("user_id","string").add("country","string").add("browser","string").add("OS","string").add("age","integer")

[6]: df=spark.createDataFrame([("A203","India","Chrome","WIN",33),("A201","China","Safari","MacOS",35), ("A205","UK","Mozilla","Linux",25)], schema=schema)

[7]: df.printSchema()

root
 |-- user_id: string (nullable = true)
 |-- country: string (nullable = true)
 |-- browser: string (nullable = true)
 |-- OS: string (nullable = true)
 |-- age: integer (nullable = true)

[8]: df.show()
```

user_id	country	browser	OS	age
A203	India	Chrome	WIN	33
A201	China	Safari	MacOS	35
A205	UK	Mozilla	Linux	25

➤ Traitement des valeurs nulles en utilisant la fonction **fillna**

```
Untitled.ipynb Python 3 (ipykernel)

[9]: df_na=spark.createDataFrame([("A203",None,"Chrome","WIN",33),("A201","China",None,"MacOS",35), ("A205","UK","Mozilla", "Linux",25)], schema=schema)

[10]: df_na.show()

+-----+-----+-----+-----+
|user_id|country|browser| OS |age|
+-----+-----+-----+-----+
| A203| null| Chrome| WIN| 33|
| A201| China| null| MacOS| 35|
| A205| UK| Mozilla| Linux| 25|
+-----+-----+-----+-----+

[11]: df_na.fillna('0').show()

+-----+-----+-----+-----+
|user_id|country|browser| OS |age|
+-----+-----+-----+-----+
| A203| 0| Chrome| WIN| 33|
| A201| China| 0| MacOS| 35|
| A205| UK| Mozilla| Linux| 25|
+-----+-----+-----+-----+

[12]: df_na.fillna({'country': 'USA', 'browser' : 'Safari'}).show()

+-----+-----+-----+-----+
|user_id|country|browser| OS |age|
+-----+-----+-----+-----+
| A203| USA| Chrome| WIN| 33|
| A201| China| Safari| MacOS| 35|
| A205| UK| Mozilla| Linux| 25|
+-----+-----+-----+-----+
```

➤ Suppression des valeurs nulles en utilisant la fonction **na.drop**

```
Untitled.ipynb Python 3 (ipykernel)

[9]: df_na=spark.createDataFrame([("A203",None,"Chrome","WIN",33),("A201","China",None,"MacOS",35), ("A205","UK","Mozilla", "Linux",25)], schema=schema)

[10]: df_na.show()

+-----+-----+-----+-----+
|user_id|country|browser| OS |age|
+-----+-----+-----+-----+
| A203| null| Chrome| WIN| 33|
| A201| China| null| MacOS| 35|
| A205| UK| Mozilla| Linux| 25|
+-----+-----+-----+-----+

[11]: df_na.na.drop().show()

+-----+-----+-----+-----+
|user_id|country|browser| OS |age|
+-----+-----+-----+-----+
| A205| UK| Mozilla| Linux| 25|
+-----+-----+-----+-----+

[12]: df_na.na.drop(subset='country').show()

+-----+-----+-----+-----+
|user_id|country|browser| OS |age|
+-----+-----+-----+-----+
| A201| China| null| MacOS| 35|
| A205| UK| Mozilla| Linux| 25|
+-----+-----+-----+-----+
```

➤ Remplacement des données par des valeurs particulières en utilisant la fonction **replace**

```
[9]: df_na=spark.createDataFrame([( "A203",None,"Chrome", "WIN",33),("A201",'China',None,"MacOS",35), ( "A205",'UK',"Mozilla", "Linux",25)], schema=schema)
```

```
[10]: df_na.show()
```

```
+-----+-----+-----+-----+
|user_id|country|browser| OS|age|
+-----+-----+-----+-----+
| A203| null| Chrome| WIN| 33|
| A201| China| null|MacOS| 35|
| A205| UK| Mozilla|Linux| 25|
+-----+-----+-----+-----+
```

```
[11]: df_na.replace("Chrome","Google Chrome").show()
```

```
+-----+-----+-----+-----+
|user_id|country| browser| OS|age|
+-----+-----+-----+-----+
| A203| null|Google Chrome| WIN| 33|
| A201| China| null|MacOS| 35|
| A205| UK| Mozilla|Linux| 25|
+-----+-----+-----+-----+
```

```
[12]: df_na.drop('user_id').show()
```

```
+-----+-----+-----+
|country|browser| OS|age|
+-----+-----+-----+
| null| Chrome| WIN| 33|
| China| null|MacOS| 35|
| UK| Mozilla|Linux| 25|
+-----+-----+-----+
```

➤ Creation d'un Dataframee Spark grace a un fichier csv

```
[9]: df=spark.read.csv("test.csv",header=True,
inferSchema=True)
```

```
[10]: df.count()
```

```
[10]: 100
```

```
[11]: len(df.columns)
```

```
[11]: 7
```

```
[12]: df.printSchema()
```

```
root
|-- name: string (nullable = true)
|-- phone: string (nullable = true)
|-- email: string (nullable = true)
|-- numberrange: integer (nullable = true)
|-- country: string (nullable = true)
|-- region: string (nullable = true)
|-- address: string (nullable = true)
```

```
[13]: df.show(3)
```

```
+-----+-----+-----+-----+-----+-----+
| name| phone| email|numberrange| country| region| address|
+-----+-----+-----+-----+-----+-----+
|Veronica Meyer|1-354-257-6683|aliquam.rutrum@ic...| 5|New Zealand| North Island|P.O. Box 359, 439...|
| Carla Ortiz|(658) 679-3482| vel.arcu@aol.com| 8| Ukraine| Sussex| 191-9609 Dui. Av.|
| Sonia Spence|(851) 587-0233|dapibus.ligula@ho...| 8| Chile|South Gyeongsang|P.O. Box 969, 995...|
+-----+-----+-----+-----+-----+-----+
only showing top 3 rows
```

```
[14]: df.summary().show()
```



summary	name	phone	email	numberrange	country	region	address
count	100	100	100	100	100	100	100
mean	null	null	null	5.57	null	null	null
stddev	null	null	null	2.91029173448264	null	null	null
min	Aidan Warren	(153) 854-5887	a@aol.ca	0	Australia	Aisén	1072 Cras Ave
25%	null	null	null	3	null	null	null
50%	null	null	null	5	null	null	null
75%	null	null	null	8	null	null	null
max	Yen Oneil	1-913-651-4439	vitae.velit@iclou...	10	Vietnam	łódzkie	P.O. Box 998, 907...

➤ Exemples d'utilisation du filtre **select**

```
[15]: df.select(['name', 'numberrange']).show()
```

name	numberrange
Veronica Meyer	5
Carla Ortiz	8
Sonia Spence	8
Dolan Bright	0
Tallulah Kim	3
Xenos Faulkner	9
Ulla Brady	9
Wendy Padilla	1
Kenyon Roman	6
Hiroko Chan	6
Elvis Rice	7
Nayda Medina	3
Denise Cortez	4
Holmes Benton	4
Penelope Franco	7
Colby Rivera	5
Pandora Sloan	10
Aline Le	9
Sybill Puckett	7
Kyla Pitts	10

only showing top 20 rows

➤ Exemples d'utilisation du filtre **Filter**

```
[16]: df.filter(df['numberrange'] > 5).count()
```

```
[16]: 49
```

```
[17]: df.filter(df['numberrange'] > 5).show()
```

name	phone	email	numberrange	country	region	address
Carla Ortiz	(658) 679-3482	vel.arcu@aol.com	8	Ukraine	Sussex	191-9609 Dui. Av.
Sonia Spence	(851) 587-0233	dapibus.ligula@ho...	8	Chile	South Gyeongsang	P.O. Box 969, 995...
Xenos Faulkner	1-913-651-4439	a@aol.ca	9	Italy	Quebec	917-2107 Enim. Road
Ulla Brady	(728) 473-5153	gravida@outlook.couk	9	Austria	Oyo	Ap #134-6792 Curs...
Kenyon Roman	(533) 323-5726	aenean@yahoo.com	6	Norway	Bali	Ap #590-1100 Lore...
Hiroko Chan	1-876-207-7590	tincidunt.nunc.ac...	6	Russian Federation	Ternopil oblast	P.O. Box 988, 860...
Elvis Rice	1-257-371-1971	maecenas.mi@hotmail...	7	Germany	Munster	501-1432 Commodore Ave
Penelope Franco	1-616-221-2177	curabitur.massa@p...	7	United Kingdom	South Jeolla	P.O. Box 642, 911...
Pandora Sloan	(813) 667-0274	fringilla.cursus@...	10	Austria	Gyeonggi	151-6092 Neque. Road
Aline Le	(274) 426-5525	ornare@aol.couk	9	Singapore	Gelderland	803-3997 Enim. Ave
Sybill Puckett	1-362-586-5515	nec@icloud.couk	7	Spain	Samara Oblast	623-2597 Fermentu...
Kyla Pitts	(708) 607-0615	dictum.mi.ac@aol.net	10	China	North Jeolla	3484 Aliquam, Street
Kiayada Buckley	1-473-461-4611	consequat.nec@pro...	9	Netherlands	Champagne-Ardenne	Ap #363-1835 Cons...
Cheyenne Bird	1-323-534-6103	tortor.nibh@proto...	7	Poland	North-East Region	778-969 Egestas Rd.
Quail Padilla	(165) 517-5136	nibh.phasellus@ya...	9	United Kingdom	Queensland	Ap #831-8639 Magn...
Maisie Ward	1-182-412-5827	pellentesque@aol.com	6	China	Brussels Hoofdste...	288-4050 Nullam St.
Carlos Guthrie	1-598-873-8483	quam.dignissim@go...	10	United States	Connacht	Ap #370-2920 Maur...
Xanthus Perez	1-174-767-0533	varius.orci.in@ou...	7	Australia	Pays de la Loire	Ap #411-2055 Curs...
Inez Mann	(647) 418-6821	ornare.in@protonm...	9	Pakistan	Limón	647-6248 Nunc Rd.
Ignatius Mclean	(436) 546-5681	integer.mollis.in...	10	Pakistan	Colorado	Ap #277-5501 Est....

only showing top 20 rows

➤ Exemples d'utilisation du GroupBy

```
df.groupby('country').count().show()
```

country	count
Sweden	2
Philippines	1
Singapore	4
Turkey	2
Germany	3
Peru	2
China	6
United States	2
Chile	4
Nigeria	1
Italy	2
Norway	2
Spain	6
Russian Federation	4
Ireland	4
Ukraine	5
South Korea	3
Mexico	2
Indonesia	5
Canada	1

only showing top 20 rows

➤ Exemples d'utilisation de l'aggregation avec les fonctions **groupBy** et **OrderBy**

```
[31]: for col in df.columns:
      if col != 'numberrange':
          print(f" Aggregation for {col}")
      df.groupBy(col).count().orderBy('count',ascending= False).show(truncate=False)
```

Aggregation for name
 Aggregation for phone
 Aggregation for email
 Aggregation for country
 Aggregation for region
 Aggregation for address

address	count
P.O. Box 900, 7515 Pretium Av.	1
486-3656 Pharetra St.	1
191-9609 Dui. Av.	1
Ap #723-5508 Consectetuer Rd.	1
Ap #937-9931 Nunc Ave	1
6846 Cursus Av.	1
Ap #480-9279 Vel Road	1
P.O. Box 359, 4399 Risus Avenue	1
933-2674 In Road	1
P.O. Box 859, 4599 Magna Av.	1
724-3232 Amet, Street	1
803-3997 Enim. Ave	1
207-7857 Tortor. St.	1
Ap #918-5811 Amet St.	1
6742 Magna. Rd.	1
Ap #370-2920 Mauris Avenue	1
505-5166 Sagittis. Road	1
377-7483 Ac St.	1
Ap #411-2055 Cursus Rd.	1
218-5176 Aliquam Av.	1

only showing top 20 rows

➤ Exemples d'utilisation de la fonction **collect_set**

```
df.groupby("country").agg(F.collect_set("numberrange")).show()
```

country	collect_set(numberrange)
Sweden	[3, 7]
Philippines	[10]
Singapore	[9, 5, 4]
Turkey	[1, 8]
Germany	[1, 7]
Peru	[0, 2]
China	[9, 6, 10, 8]
United States	[1, 10]
Chile	[1, 5, 10, 8]
Nigeria	[4]
Italy	[0, 9]
Norway	[6, 4]
Spain	[2, 7, 4, 8]
Russian Federation	[1, 2, 6, 10]
Ireland	[5, 2, 6, 7]
Ukraine	[9, 5, 3, 4, 8]
South Korea	[9, 3, 8]
Mexico	[9, 4]
Indonesia	[9, 2, 7, 4, 8]
Canada	[0]

only showing top 20 rows

- Exemples d'ajout d'une colonne avec une valeur constante grace a la fonction **lit**

```
[36]: df=df.withColumn('constant',F.lit('finance'))
```

```
[37]: df.select('country','constant').show()
```

```
+-----+-----+
|      country|constant|
+-----+-----+
|   New Zealand|finance|
|      Ukraine|finance|
|        Chile|finance|
|        Peru |finance|
|United Kingdom|finance|
|        Italy|finance|
|      Austria|finance|
|      Vietnam|finance|
|      Norway |finance|
|Russian Federation|finance|
|      Germany|finance|
|    Costa Rica|finance|
|   New Zealand|finance|
|   Netherlands|finance|
|United Kingdom|finance|
|      Singapore|finance|
|      Austria |finance|
|      Singapore|finance|
|        Spain |finance|
|        China |finance|
+-----+-----+
only showing top 20 rows
```

- Exemples de creation d'un objet **UDF**

```
[38]: from pyspark.sql.functions import udf
```

```
[39]: df.groupby("numberrange").count().show()
```

numberrange	count
1	6
6	6
3	8
5	10
9	14
4	16
8	10
7	11
10	8
2	7
0	4

```
[48]: def numberrange_category(numberrange):  
    if numberrange == 6:  
        return "Young"  
    elif numberrange == 7:  
        return "Mid Aged"  
    elif ((numberrange == 8) or (numberrange == 9)):  
        return "Old"  
    else:  
        return "Very Old"
```

```
[49]: numberrange_udf=udf(numberrange_category,StringType())
```

```
[50]: df=df.withColumn('numberrange_category',numberrange_udf(df['numberrange']))
```

```
[51]: df.select('numberrange','numberrange_category').show()
```

numberrange	numberrange_category
5	Very Old
8	Old
8	Old
0	Very Old
3	Very Old
9	Old
9	Old
1	Very Old
6	Young
6	Young
7	Mid Aged
3	Very Old
4	Very Old
4	Very Old
7	Mid Aged
5	Very Old
10	Very Old
9	Old
7	Mid Aged
10	Very Old

only showing top 20 rows

```
[52]: df.groupby("numberrange_category").count().show()
```

```
+-----+-----+
|numberrange_category|count|
+-----+-----+
|          Mid Aged|    11|
|          Very Old|    59|
|              Old|    24|
|          Young|     6|
+-----+-----+
```

➤ Exemple d'utilisation de la fonction sort

```
[33]: df.sort("numberrange", ascending=False).show()
```

```
+-----+-----+-----+-----+-----+-----+-----+
|name|phone|email|numberrange|country|region|address|
+-----+-----+-----+-----+-----+-----+-----+
|Alyssa Flowers|(474) 814-5408|ut.nisi@google.net|10|United Kingdom|Alajuela|Ap #395-9214 Proi...|
|Aquila Ramsey|1-778-832-1156|adipiscing@google...|10|Russian Federation|Xibēi|182-7995 Eu, Street|
|Carlos Guthrie|1-598-873-8483|quam.dignissim@go...|10|United States|Connacht|Ap #370-2920 Maur...|
|Angela Mcbride|1-847-365-8558|non.massa@aol.couk|10|Chile|Gangwon|Ap #830-721 Ac St.|
|Sebastian Hyde|1-272-511-5727|lorem@hotmail.edu|10|Philippines|Lower Austria|496-8909 Ipsum Av...|
|Kyla Pitts|(708) 607-0615|dictum.mi.ac@aol.net|10|China|North Jeolla|3484 Aliquam, Street|
|Ignatius Mclean|(436) 546-5681|integer.mollis.in...|10|Pakistan|Colorado|Ap #277-5501 Est....|
|Pandora Sloan|(813) 667-0274|fringilla.cursus@...|10|Austria|Gyeonggi|151-6092 Neque. Road|
|Inez Rocha|(901) 759-8291|et@aol.org|9|United Kingdom|Western Australia|Ap #623-5127 Ulla...|
|Xenos Faulkner|1-913-651-4439|a@aol.ca|9|Italy|Quebec|917-2107 Enim. Road|
|Inez Mann|(647) 418-6821|ornare.in@protonm...|9|Pakistan|Limón|647-6248 Nunc Rd.|
|Aline Le|(274) 426-5525|ornare@aol.couk|9|Singapore|Gelderland|803-3997 Enim. Ave|
|Quail Padilla|(165) 517-5136|nibh.phasellus@ya...|9|United Kingdom|Queensland|Ap #831-8639 Magn...|
|Iris Baird|1-401-376-4755|tempus@yahoo.ca|9|Ukraine|Berlin|Ap #260-9030 Sapi...|
|Kiayada Buckley|1-473-461-4611|consequat.nec@pro...|9|Netherlands|Champagne-Ardenne|Ap #363-1835 Cons...|
|Chantale Henson|1-153-527-0712|molestie@protonma...|9|Indonesia|Central Java|Ap #562-4144 Eget...|
|Nyssa Coffey|1-745-548-6706|lacinia.vitae@yah...|9|South Korea|Bremen|P.O. Box 900, 751...|
|Sacha Church|(574) 888-8424|sodales.mauris.bl...|9|Pakistan|Guanajuato|Ap #784-4557 Frin...|
|Ulla Brady|(728) 473-5153|gravida@outlook.couk|9|Austria|Oyo|Ap #134-6792 Curs...|
|Samson Clements|1-541-258-7647|rutrum.eu@google.ca|9|China|Luxemburg|P.O. Box 467, 127...|
+-----+-----+-----+-----+-----+-----+-----+
```

only showing top 20 rows