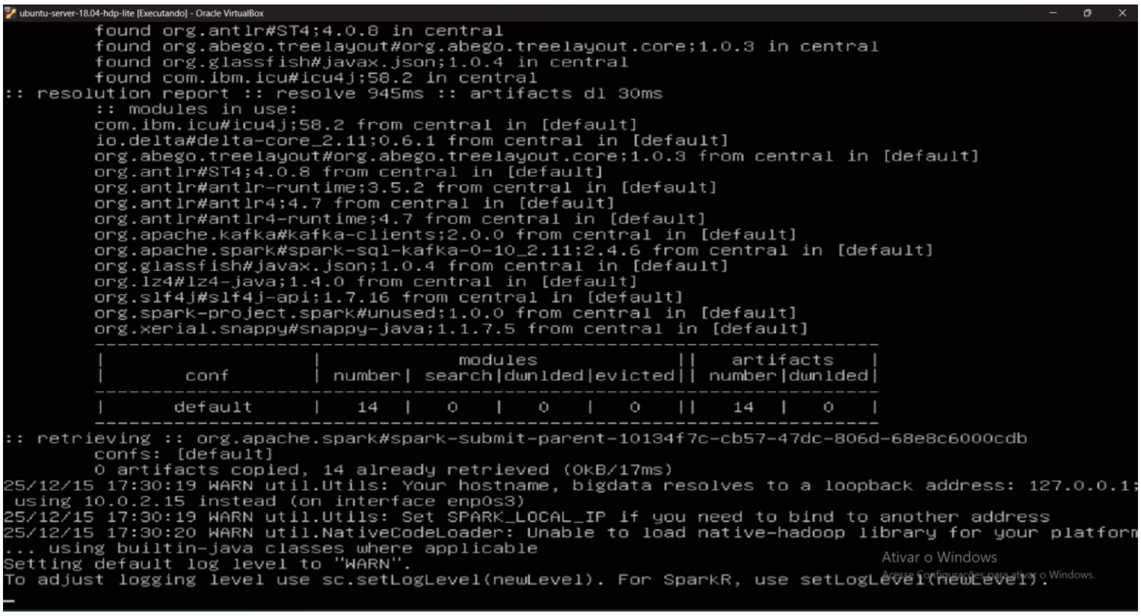
Rodando o 1-postgres2kafka

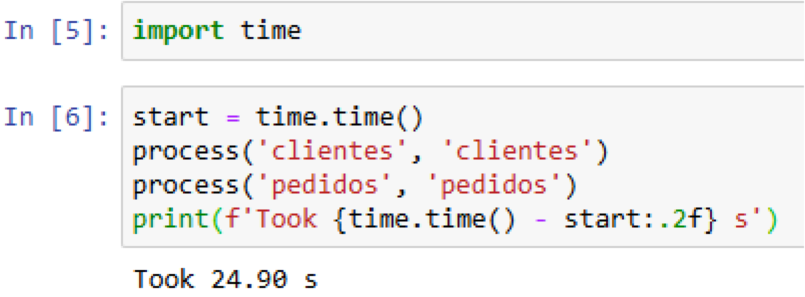
Lendo os dados do Postgres e publicando no Kafka

Arquivo *config.yaml* tem os dados de acesso ao Postgres e Kafka:


ubuntu-server-18.04-hdp-lite [Executando] - Oracle VirtualBox
Ativar o Windows
Acesse Configurações para ativar o Windows.

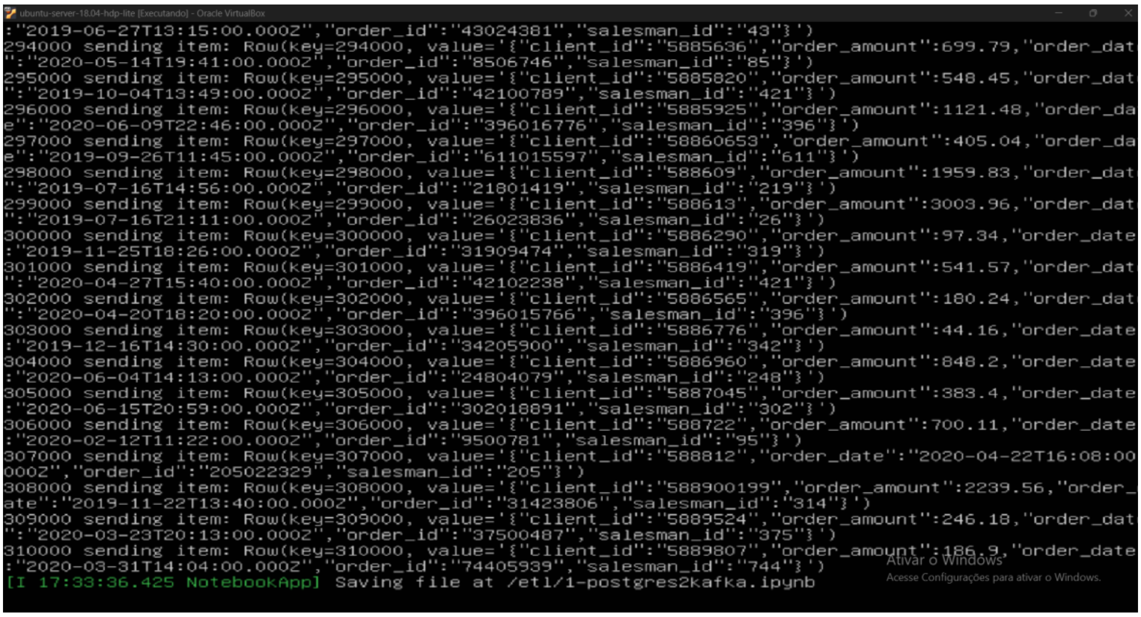
```
        found org.antlr#ST4;4.0.8 in central
        found org.abego.treelayout#org.abego.treelayout.core;1.0.3 in central
        found org.glassfish#javax.json;1.0.4 in central
        found com.ibm.icu#icu4j;58.2 in central
:: resolution report :: resolve 945ms :: artifacts dl 30ms
        :: modules in use:
        com.ibm.icu#icu4j;58.2 from central in [default]
        io.delta#delta-core_2.11;0.6.1 from central in [default]
        org.abego.treelayout#org.abego.treelayout.core;1.0.3 from central in [default]
        org.antlr#ST4;4.0.8 from central in [default]
        org.antlr#antlr-runtime;3.5.2 from central in [default]
        org.antlr#antlr4;4.7 from central in [default]
        org.antlr#antlr4-runtime;4.7 from central in [default]
        org.apache.kafka#kafka-clients;2.0.0 from central in [default]
        org.apache.spark#spark-sql-kafka-0-10_2.11;2.4.6 from central in [default]
        org.glassfish#javax.json;1.0.4 from central in [default]
        org.lz4#lz4-java;1.4.0 from central in [default]
        org.slf4j#slf4j-api;1.7.16 from central in [default]
        org.spark-project.spark#unused;1.0.0 from central in [default]
        org.xerial.snappy#snappy-java;1.1.7.5 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   14  |   0   |   0   |   0   ||   14  |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-10134f7c-cb57-47dc-806d-68e8c6000cdb
        confs: [default]
        0 artifacts copied, 14 already retrieved (0kB/17ms)
25/12/15 17:30:19 WARN util.Utils: Your hostname, bigdata resolves to a loopback address: 127.0.0.1;
 using 10.0.2.15 instead (on interface enp0s3)
25/12/15 17:30:19 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another address
25/12/15 17:30:20 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform
... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
_
```

Leitura do Postgres e escrita no Kafka:



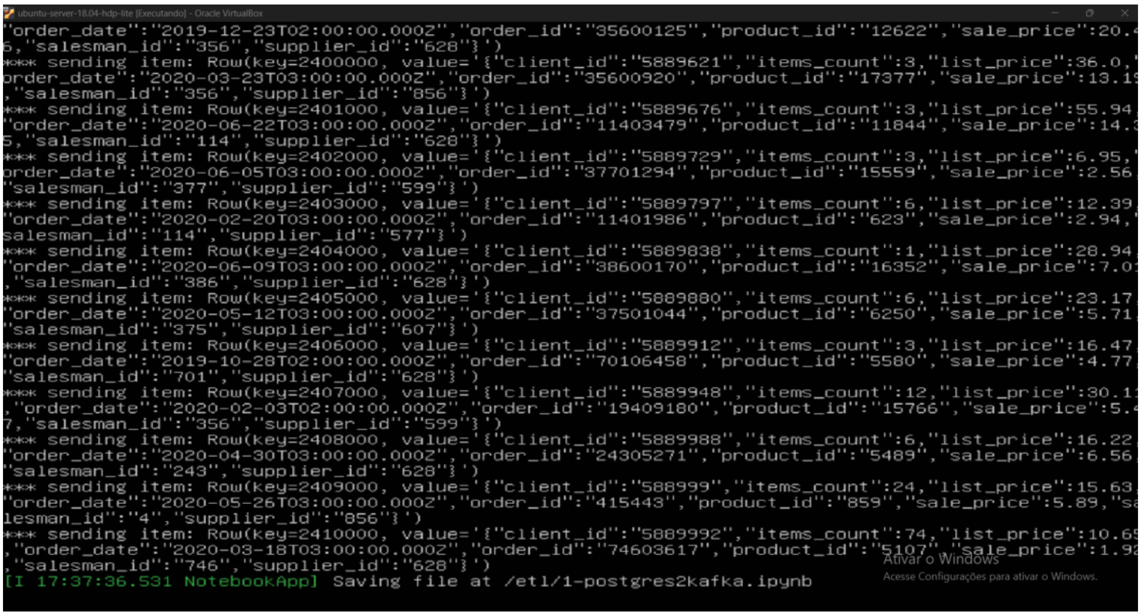
```
In [5]: import time
```

```
In [6]: start = time.time()
        process('clientes', 'clientes')
        process('pedidos', 'pedidos')
        print(f'Took {time.time() - start:.2f} s')

        Took 24.90 s
```


ubuntu-server-18.04-hdp-lite [Executando] - Oracle VirtualBox
Ativar o Windows
Acesse Configurações para ativar o Windows.

```
:"2019-06-27T13:15:00.000Z","order_id":"43024381","salesman_id":"43"}')
294000 sending item: Row(key=294000, value='{"client_id":"5885636","order_amount":699.79,"order_dat
":"2020-05-14T19:41:00.000Z","order_id":"8506746","salesman_id":"85"}')
295000 sending item: Row(key=295000, value='{"client_id":"5885820","order_amount":548.45,"order_dat
":"2019-10-04T13:49:00.000Z","order_id":"42100789","salesman_id":"421"}')
296000 sending item: Row(key=296000, value='{"client_id":"5885925","order_amount":1121.48,"order_da
e":"2020-06-09T22:46:00.000Z","order_id":"396016776","salesman_id":"396"}')
297000 sending item: Row(key=297000, value='{"client_id":"58860653","order_amount":405.04,"order_da
e":"2019-09-26T11:45:00.000Z","order_id":"611015597","salesman_id":"611"}')
298000 sending item: Row(key=298000, value='{"client_id":"588609","order_amount":1959.83,"order_dat
":"2019-07-16T14:56:00.000Z","order_id":"21801419","salesman_id":"219"}')
299000 sending item: Row(key=299000, value='{"client_id":"588613","order_amount":3003.96,"order_dat
":"2019-07-16T21:11:00.000Z","order_id":"26023836","salesman_id":"26"}')
300000 sending item: Row(key=300000, value='{"client_id":"5886290","order_amount":97.34,"order_date
:"2019-11-25T18:26:00.000Z","order_id":"31909474","salesman_id":"319"}')
301000 sending item: Row(key=301000, value='{"client_id":"5886419","order_amount":541.57,"order_dat
":"2020-04-27T15:40:00.000Z","order_id":"42102238","salesman_id":"421"}')
302000 sending item: Row(key=302000, value='{"client_id":"5886565","order_amount":180.24,"order_dat
":"2020-04-20T18:20:00.000Z","order_id":"396015766","salesman_id":"396"}')
303000 sending item: Row(key=303000, value='{"client_id":"5886776","order_amount":44.16,"order_date
:"2019-12-16T14:30:00.000Z","order_id":"34205900","salesman_id":"342"}')
304000 sending item: Row(key=304000, value='{"client_id":"5886960","order_amount":848.2,"order_date
:"2020-06-04T14:13:00.000Z","order_id":"24804079","salesman_id":"248"}')
305000 sending item: Row(key=305000, value='{"client_id":"5887045","order_amount":383.4,"order_date
:"2020-06-15T20:59:00.000Z","order_id":"302018891","salesman_id":"302"}')
306000 sending item: Row(key=306000, value='{"client_id":"588722","order_amount":700.11,"order_date
:"2020-02-12T11:22:00.000Z","order_id":"9500781","salesman_id":"95"}')
307000 sending item: Row(key=307000, value='{"client_id":"588812","order_date":"2020-04-22T16:08:00
000Z","order_id":"205022329","salesman_id":"205"}')
308000 sending item: Row(key=308000, value='{"client_id":"588900199","order_amount":2239.56,"order_
ate":"2019-11-22T13:40:00.000Z","order_id":"31423806","salesman_id":"314"}')
309000 sending item: Row(key=309000, value='{"client_id":"5889524","order_amount":246.18,"order_dat
":"2020-03-23T20:13:00.000Z","order_id":"37500487","salesman_id":"375"}')
310000 sending item: Row(key=310000, value='{"client_id":"5889807","order_amount":186.9,"order_date
:"2020-03-31T14:04:00.000Z","order_id":"74405939","salesman_id":"744"}')
[I 17:33:36.425 NotebookApp] Saving file at /etl/1-postgres2kafka.ipynb
```

Percorrendo as partições do Dataframe df_itens, enviando os dados para o Kafka, chamando a função send_itens():

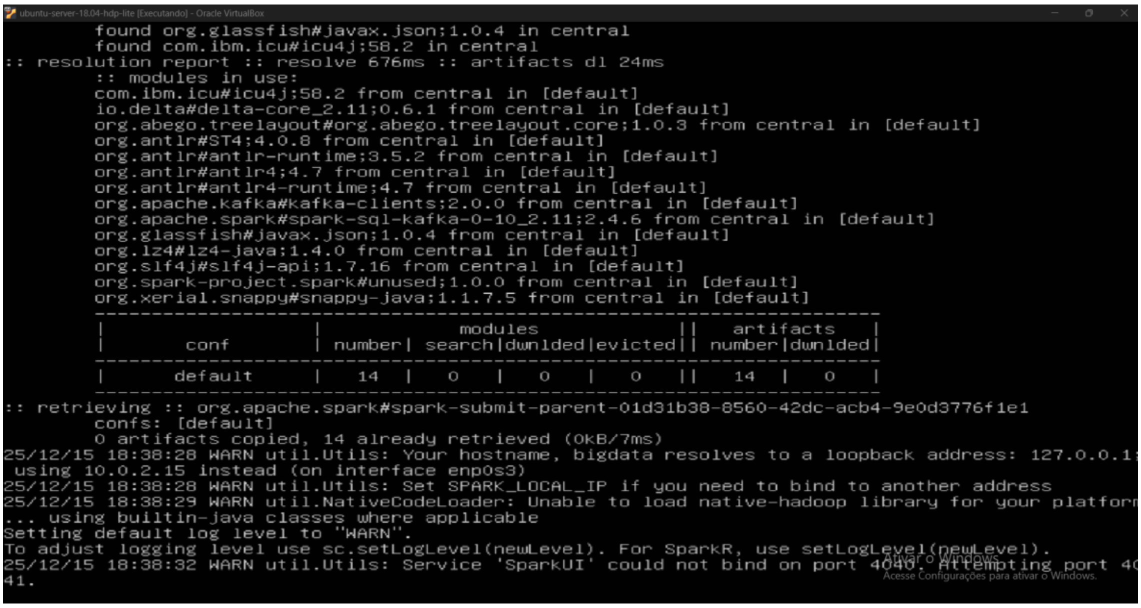

ubuntu-server-18.04-hdp-lite [Executando] - Oracle VirtualBox

```
"order_date":"2019-12-23T02:00:00.000Z","order_id":"35600125","product_id":"12622","sale_price":20.4
6,"salesman_id":"356","supplier_id":"628"}')
*** sending item: Row(key=2400000, value='{"client_id":"5889621","items_count":3,"list_price":36.0,"
order_date":"2020-03-23T03:00:00.000Z","order_id":"35600920","product_id":"17377","sale_price":13.19
,"salesman_id":"356","supplier_id":"856"}')
*** sending item: Row(key=2401000, value='{"client_id":"5889676","items_count":3,"list_price":55.94,
"order_date":"2020-06-22T03:00:00.000Z","order_id":"11403479","product_id":"11844","sale_price":14.3
5,"salesman_id":"114","supplier_id":"628"}')
*** sending item: Row(key=2402000, value='{"client_id":"5889729","items_count":3,"list_price":6.95,"
order_date":"2020-06-05T03:00:00.000Z","order_id":"37701294","product_id":"15559","sale_price":2.56,
"salesman_id":"377","supplier_id":"599"}')
*** sending item: Row(key=2403000, value='{"client_id":"5889797","items_count":6,"list_price":12.39,
"order_date":"2020-02-20T03:00:00.000Z","order_id":"11401986","product_id":"623","sale_price":2.94,"
salesman_id":"114","supplier_id":"577"}')
*** sending item: Row(key=2404000, value='{"client_id":"5889838","items_count":1,"list_price":28.94,
"order_date":"2020-06-09T03:00:00.000Z","order_id":"38600170","product_id":"16352","sale_price":7.01
,"salesman_id":"386","supplier_id":"628"}')
*** sending item: Row(key=2405000, value='{"client_id":"5889880","items_count":6,"list_price":23.17,
"order_date":"2020-05-12T03:00:00.000Z","order_id":"37501044","product_id":"6250","sale_price":5.71,
"salesman_id":"375","supplier_id":"607"}')
*** sending item: Row(key=2406000, value='{"client_id":"5889912","items_count":3,"list_price":16.47,
"order_date":"2019-10-28T02:00:00.000Z","order_id":"70106458","product_id":"5580","sale_price":4.77,
"salesman_id":"701","supplier_id":"628"}')
*** sending item: Row(key=2407000, value='{"client_id":"5889948","items_count":12,"list_price":30.11
,"order_date":"2020-02-03T02:00:00.000Z","order_id":"19409180","product_id":"15766","sale_price":5.4
7,"salesman_id":"356","supplier_id":"599"}')
*** sending item: Row(key=2408000, value='{"client_id":"5889988","items_count":6,"list_price":16.22,
"order_date":"2020-04-30T03:00:00.000Z","order_id":"24305271","product_id":"5489","sale_price":6.56,
"salesman_id":"243","supplier_id":"628"}')
*** sending item: Row(key=2409000, value='{"client_id":"588999","items_count":24,"list_price":15.63,
"order_date":"2020-05-26T03:00:00.000Z","order_id":"415443","product_id":"859","sale_price":5.89,"sa
lesman_id":"4","supplier_id":"856"}')
*** sending item: Row(key=2410000, value='{"client_id":"5889992","items_count":74,"list_price":10.65
,"order_date":"2020-03-18T03:00:00.000Z","order_id":"74603617","product_id":"5107","sale_price":1.92
,"salesman_id":"746","supplier_id":"628"}')
[I 17:37:36.531 NotebookApp] Saving file at /etl/1-postgres2kafka.ipynb
```

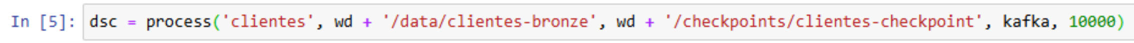
Rodando o 2-kafka2bronze

Movendo os dados do Kafka para o Delta

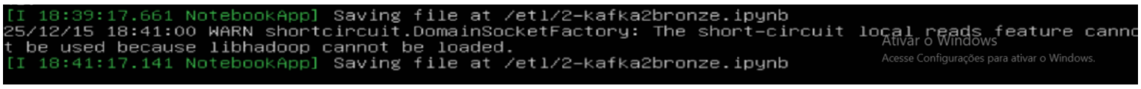
Arquivo config.yaml tem os dados de acesso ao Postgres e Kafka.


ubuntu-server-18.04-hdp-lite [Executando] - Oracle VirtualBox

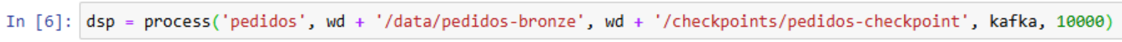
```
        found org.glassfish#javax.json;1.0.4 in central
        found com.ibm.icu#icu4j;58.2 in central
:: resolution report :: resolve 676ms :: artifacts dl 24ms
        :: modules in use:
        com.ibm.icu#icu4j;58.2 from central in [default]
        io.delta#delta-core_2.11;0.6.1 from central in [default]
        org.abego.treelayout#org.abego.treelayout.core;1.0.3 from central in [default]
        org.antlr#ST4;4.0.8 from central in [default]
        org.antlr#antlr-runtime;3.5.2 from central in [default]
        org.antlr#antlr4;4.7 from central in [default]
        org.antlr#antlr4-runtime;4.7 from central in [default]
        org.apache.kafka#kafka-clients;2.0.0 from central in [default]
        org.apache.spark#spark-sql-kafka-0-10_2.11;2.4.6 from central in [default]
        org.glassfish#javax.json;1.0.4 from central in [default]
        org.lz4#lz4-java;1.4.0 from central in [default]
        org.slf4j#slf4j-api;1.7.16 from central in [default]
        org.spark-project.spark#unused;1.0.0 from central in [default]
        org.xerial.snappy#snappy-java;1.1.7.5 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   14  |   0   |   0   |   0   ||   14  |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-01d31b38-8560-42dc-acb4-9e0d3776f1e1
        confs: [default]
        0 artifacts copied, 14 already retrieved (0kB/7ms)
25/12/15 18:38:28 WARN util.Utils: Your hostname, bigdata resolves to a loopback address: 127.0.0.1;
 using 10.0.2.15 instead (on interface enp0s3)
25/12/15 18:38:28 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another address
25/12/15 18:38:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform
... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/12/15 18:38:32 WARN util.Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 40
41.
```



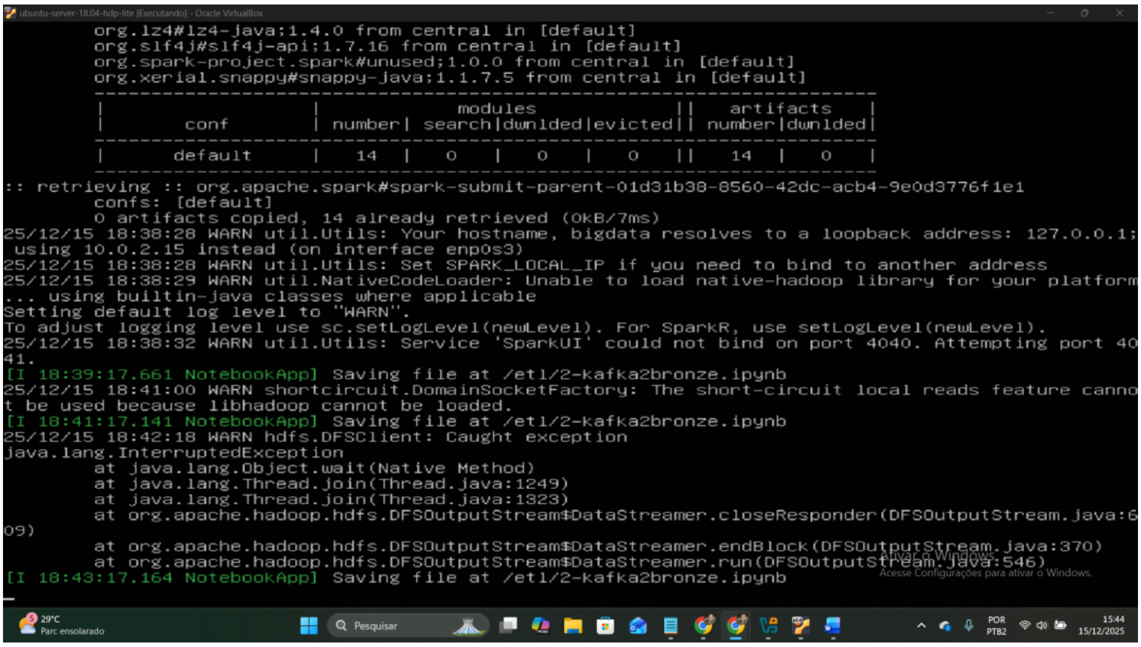
```
In [5]: dsc = process('clientes', wd + '/data/clientes-bronze', wd + '/checkpoints/clientes-checkpoint', kafka, 10000)
```



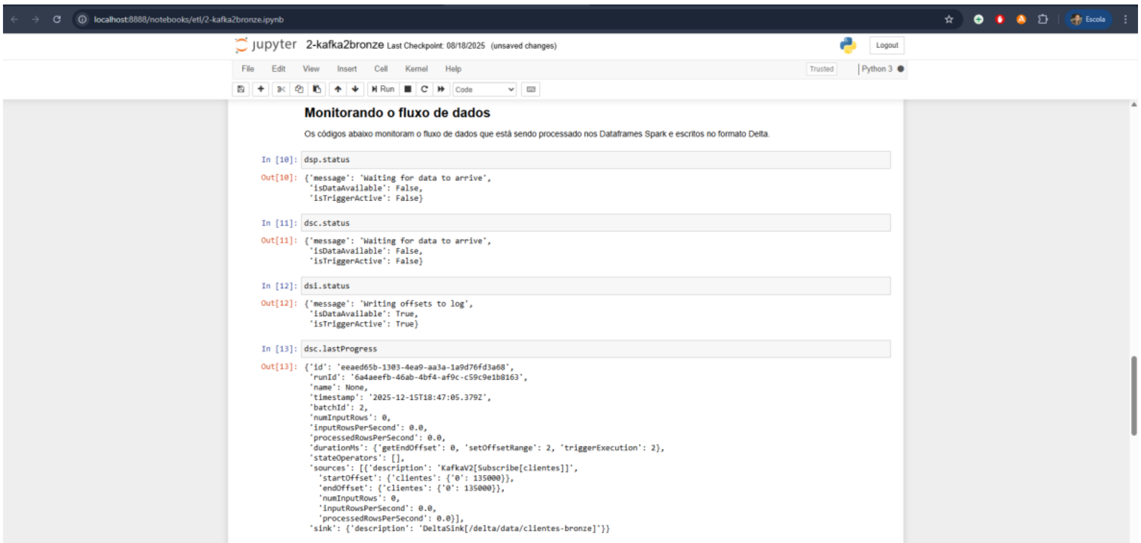
```
[I 18:39:17.661 NotebookApp] Saving file at /etl/2-kafka2bronze.ipynb
25/12/15 18:41:00 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature canno
t be used because libhadoop cannot be loaded.
[I 18:41:17.141 NotebookApp] Saving file at /etl/2-kafka2bronze.ipynb
```



```
In [6]: dsp = process('pedidos', wd + '/data/pedidos-bronze', wd + '/checkpoints/pedidos-checkpoint', kafka, 10000)
```


ubuntu-server-18.04-hdp-lite [Executando] - Oracle VirtualBox
29°C
Parc ensolarado
Pesquisar
POR
PTB2
15:44
15/12/2025

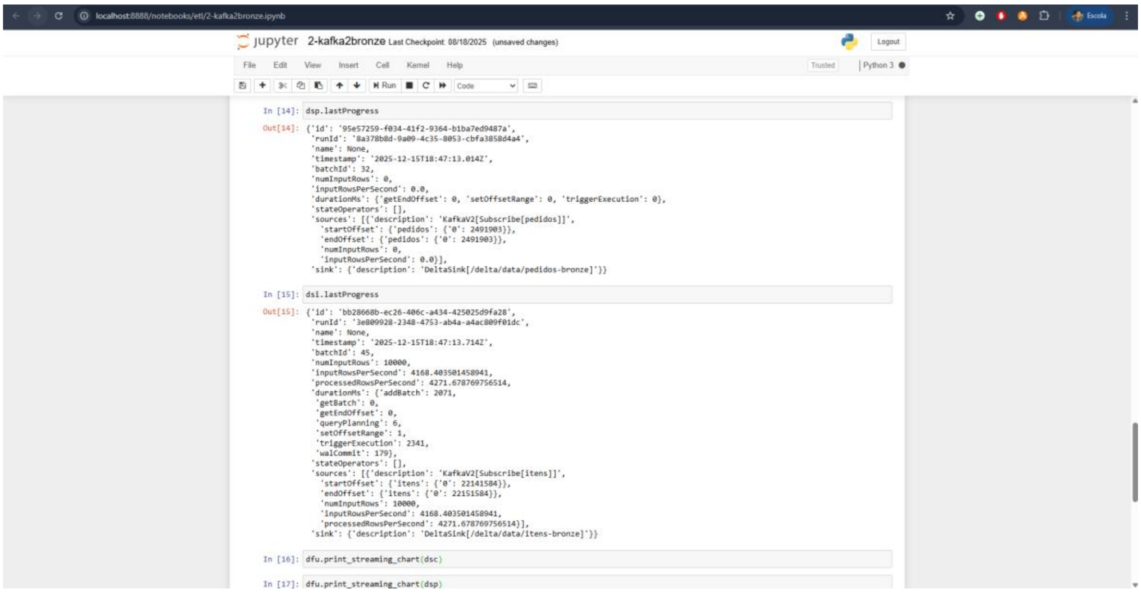
```
        org.lz4#lz4-java;1.4.0 from central in [default]
        org.slf4j#slf4j-api;1.7.16 from central in [default]
        org.spark-project.spark#unused;1.0.0 from central in [default]
        org.xerial.snappy#snappy-java;1.1.7.5 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   14  |   0   |   0   |   0   ||   14  |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-01d31b38-8560-42dc-acb4-9e0d3776f1e1
        confs: [default]
        0 artifacts copied, 14 already retrieved (0kB/7ms)
25/12/15 18:38:28 WARN util.Utils: Your hostname, bigdata resolves to a loopback address: 127.0.0.1;
 using 10.0.2.15 instead (on interface enp0s3)
25/12/15 18:38:28 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another address
25/12/15 18:38:29 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform
... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/12/15 18:38:32 WARN util.Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 40
41.
[I 18:39:17.661 NotebookApp] Saving file at /etl/2-kafka2bronze.ipynb
25/12/15 18:41:00 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature canno
t be used because libhadoop cannot be loaded.
[I 18:41:17.141 NotebookApp] Saving file at /etl/2-kafka2bronze.ipynb
25/12/15 18:42:18 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1249)
        at java.lang.Thread.join(Thread.java:1323)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:6
09)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:370)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:546)
[I 18:43:17.164 NotebookApp] Saving file at /etl/2-kafka2bronze.ipynb
_
```


localhost:8888/notebooks/etl/2-kafka2bronze.ipynb
jupyter
2-kafka2bronze Last Checkpoint: 08/18/2025 (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Trusted
Python 3
Run
Code

Monitorando o fluxo de dados

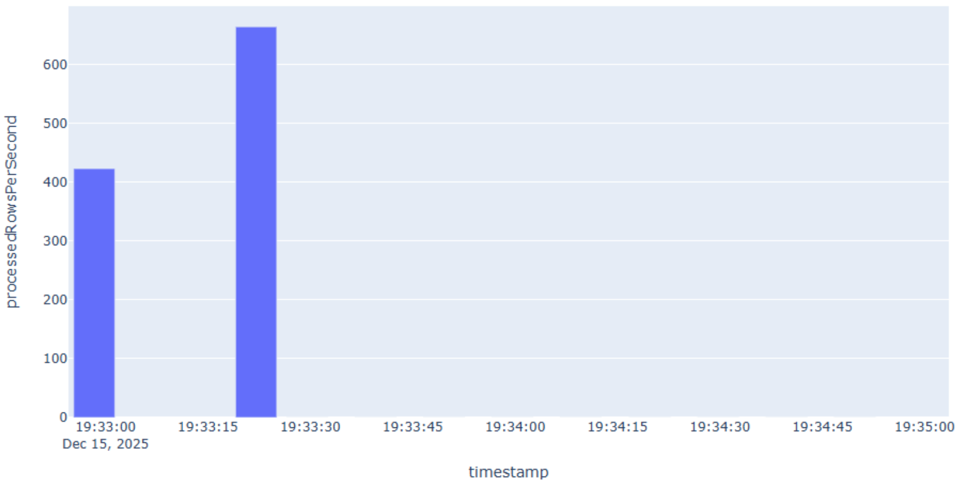
Os códigos abaixo monitoram o fluxo de dados que está sendo processado nos Dataframes Spark e escritos no formato Delta.

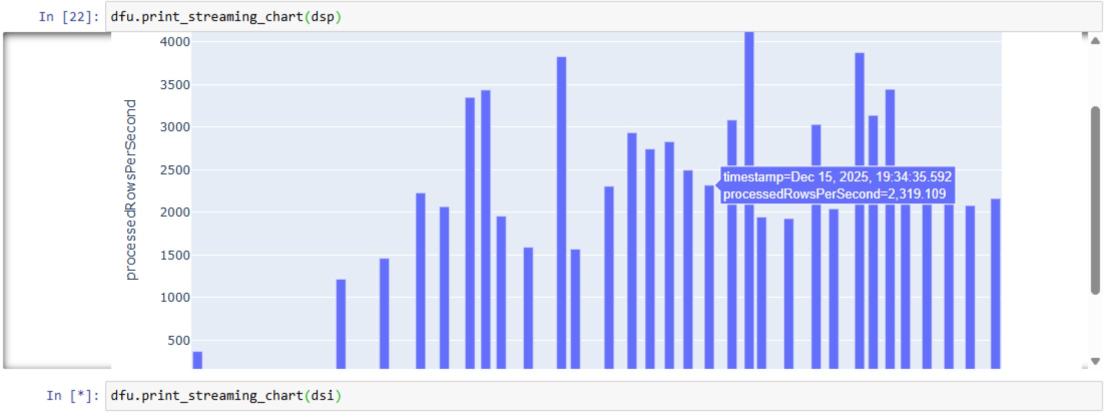
```
In [10]: dsp.status

Out[10]: {'message': 'Waiting for data to arrive',
          'isDataAvailable': False,
          'isTriggerActive': False}

In [11]: dsc.status

Out[11]: {'message': 'Waiting for data to arrive',
          'isDataAvailable': False,
          'isTriggerActive': False}

In [12]: dsi.status

Out[12]: {'message': 'Writing offsets to log',
          'isDataAvailable': True,
          'isTriggerActive': True}

In [13]: dsc.lastProgress

Out[13]: {'id': 'eeaed65b-1303-4ea9-aa3a-1a9d76fd3a68',
          'runId': '6a4aeefb-46ab-4bf4-af9c-c59c9e1b8163',
          'name': None,
          'timestamp': '2025-12-15T18:47:05.379Z',
          'batchId': 2,
          'numInputRows': 0,
          'inputRowsPerSecond': 0.0,
          'processedRowsPerSecond': 0.0,
          'durationMs': {'getEndOffset': 0, 'setOffsetRange': 2, 'triggerExecution': 2},
          'stateOperators': [],
          'sources': [{'description': 'KafkaV2[Subscribe[clientes]]',
            'startOffset': {'clientes': {'0': 135000}},
            'endOffset': {'clientes': {'0': 135000}},
            'numInputRows': 0,
            'inputRowsPerSecond': 0.0,
            'processedRowsPerSecond': 0.0}],
          'sink': {'description': 'DeltaSink[/delta/data/clientes-bronze]'}}
```


localhost:8888/notebooks/etl/2-kafka2bronze.ipynb
jupyter
2-kafka2bronze Last Checkpoint: 08/18/2025 (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Trusted
Python 3
Run
Code

```
In [14]: dsp.lastProgress

Out[14]: {'id': '95e57259-f034-41f2-9364-b1ba7ed9487a',
          'runId': '8a378b8d-9a09-4c35-8053-cbfa3858d4a4',
          'name': None,
          'timestamp': '2025-12-15T18:47:13.014Z',
          'batchId': 32,
          'numInputRows': 0,
          'inputRowsPerSecond': 0.0,
          'durationMs': {'getEndOffset': 0, 'setOffsetRange': 0, 'triggerExecution': 0},
          'stateOperators': [],
          'sources': [{'description': 'KafkaV2[Subscribe[pedidos]]',
            'startOffset': {'pedidos': {'0': 2491903}},
            'endOffset': {'pedidos': {'0': 2491903}},
            'numInputRows': 0,
            'inputRowsPerSecond': 0.0}],
          'sink': {'description': 'DeltaSink[/delta/data/pedidos-bronze]'}}

In [15]: dsi.lastProgress

Out[15]: {'id': 'bb28668b-ec26-406c-a434-425025d9fa28',
          'runId': '3e809928-2348-4753-ab4a-a4ac809f01dc',
          'name': None,
          'timestamp': '2025-12-15T18:47:13.714Z',
          'batchId': 45,
          'numInputRows': 10000,
          'inputRowsPerSecond': 4168.403501458941,
          'processedRowsPerSecond': 4271.678769756514,
          'durationMs': {'addBatch': 2071,
           'getBatch': 0,
           'getEndOffset': 0,
           'queryPlanning': 6,
           'setOffsetRange': 1,
           'triggerExecution': 2341,
           'walCommit': 179},
          'stateOperators': [],
          'sources': [{'description': 'KafkaV2[Subscribe[itens]]',
            'startOffset': {'itens': {'0': 22141584}},
            'endOffset': {'itens': {'0': 22151584}},
            'numInputRows': 10000,
            'inputRowsPerSecond': 4168.403501458941,
            'processedRowsPerSecond': 4271.678769756514}],
          'sink': {'description': 'DeltaSink[/delta/data/itens-bronze]'}}

In [16]: dfu.print_streaming_chart(dsc)

In [17]: dfu.print_streaming_chart(dsp)
```

```
In [21]: dfu.print_streaming_chart(dsc)
```

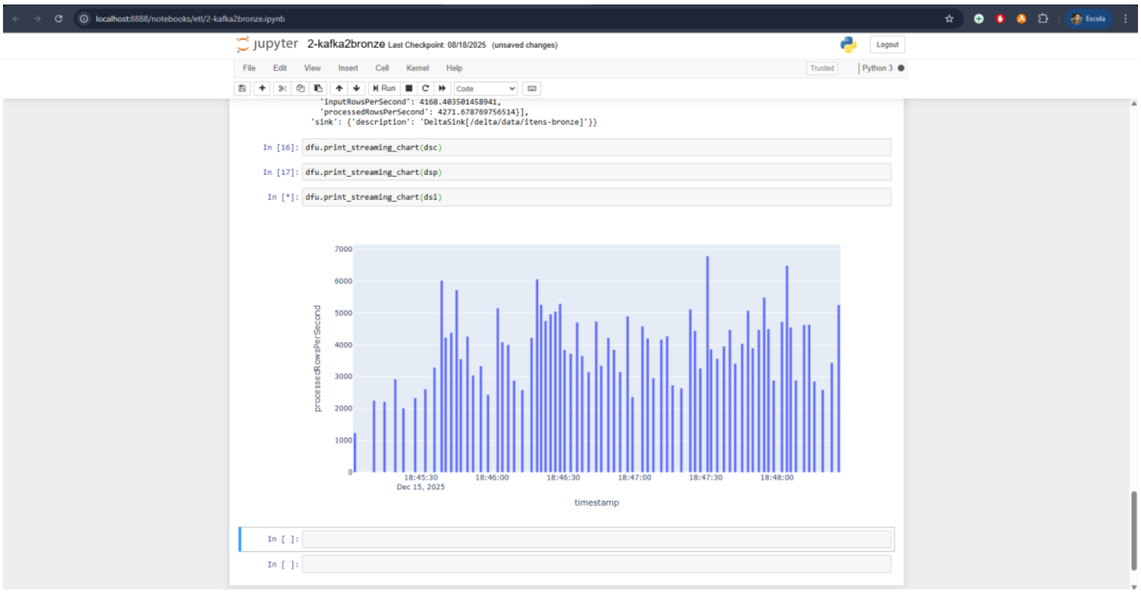




```
In [22]: dfu.print_streaming_chart(dsp)
```

4000
3500
3000
2500
2000
1500
1000
500
processedRowsPerSecond
timestamp=Dec 15, 2025, 19:34:35.592
processedRowsPerSecond=2,319.109

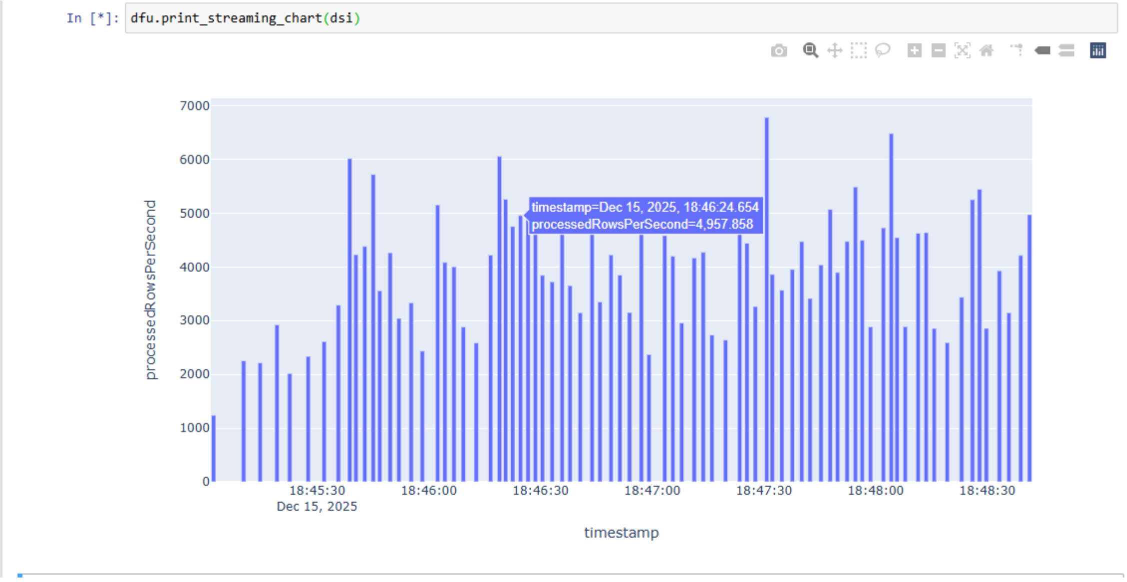
```
In [*]: dfu.print_streaming_chart(dsi)
```



localhost:8888/notebooks/etl/2-kafka2bronze.ipynb
Escola
jupyter
2-kafka2bronze Last Checkpoint: 08/18/2025 (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Trusted
Python 3
Run
Code

```
  'inputRowsPerSecond': 4168.403501458941,
  'processedRowsPerSecond': 4271.678769756514}],
 'sink': {'description': 'DeltaSink[/delta/data/itens-bronze]'}}

In [16]: dfu.print_streaming_chart(dsc)

In [17]: dfu.print_streaming_chart(dsp)

In [*]: dfu.print_streaming_chart(dsi)
```

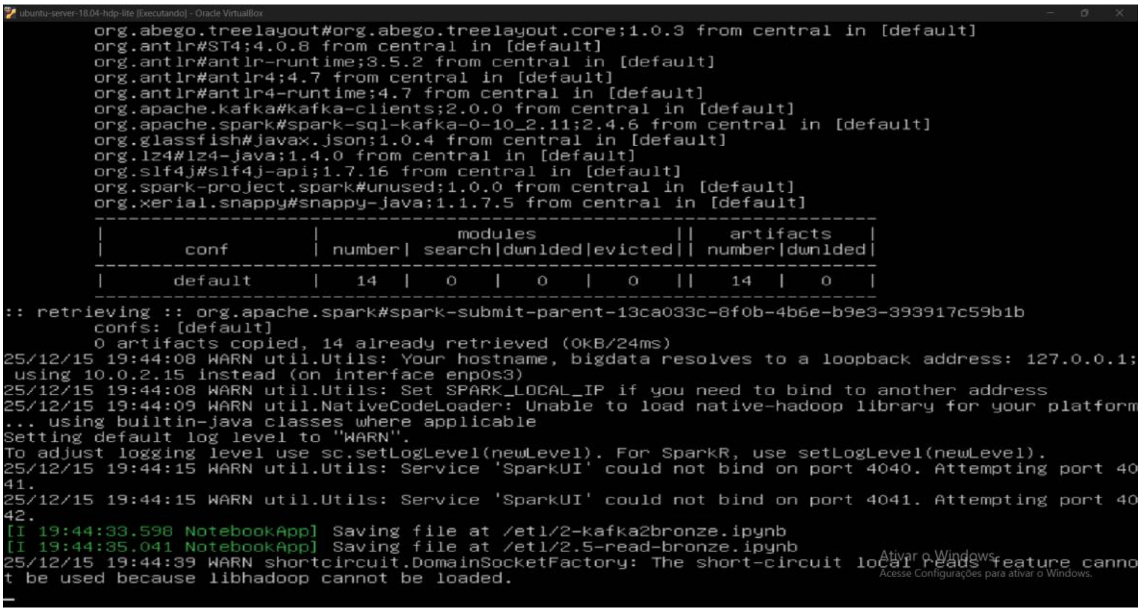
```
In [ ]:

In [ ]:
```



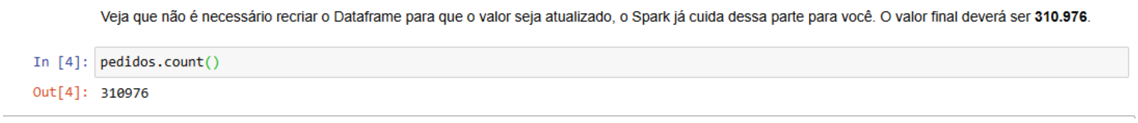
```
In [*]: dfu.print_streaming_chart(dsi)
```

Rodando 2.5-read-bronze

Arquivo config.yaml tem os dados de acesso ao Postgres e Kafka:

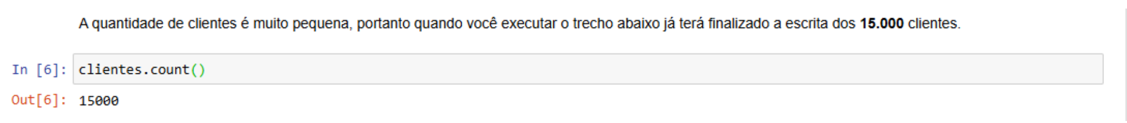

ubuntu-server-18.04-hdp-lite [Executando] - Oracle VirtualBox

```
        org.abego.treelayout#org.abego.treelayout.core;1.0.3 from central in [default]
        org.antlr#ST4;4.0.8 from central in [default]
        org.antlr#antlr-runtime;3.5.2 from central in [default]
        org.antlr#antlr4;4.7 from central in [default]
        org.antlr#antlr4-runtime;4.7 from central in [default]
        org.apache.kafka#kafka-clients;2.0.0 from central in [default]
        org.apache.spark#spark-sql-kafka-0-10_2.11;2.4.6 from central in [default]
        org.glassfish#javax.json;1.0.4 from central in [default]
        org.lz4#lz4-java;1.4.0 from central in [default]
        org.slf4j#slf4j-api;1.7.16 from central in [default]
        org.spark-project.spark#unused;1.0.0 from central in [default]
        org.xerial.snappy#snappy-java;1.1.7.5 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   14  |   0   |   0   |   0   ||   14  |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-13ca033c-8f0b-4b6e-b9e3-393917c59b1b
        confs: [default]
        0 artifacts copied, 14 already retrieved (0kB/24ms)
25/12/15 19:44:08 WARN util.Utils: Your hostname, bigdata resolves to a loopback address: 127.0.0.1;
 using 10.0.2.15 instead (on interface enp0s3)
25/12/15 19:44:08 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another address
25/12/15 19:44:09 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform
... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/12/15 19:44:15 WARN util.Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 40
41.
25/12/15 19:44:15 WARN util.Utils: Service 'SparkUI' could not bind on port 4041. Attempting port 40
42.
[I 19:44:33.598 NotebookApp] Saving file at /etl/2-kafka2bronze.ipynb
[I 19:44:35.041 NotebookApp] Saving file at /etl/2.5-read-bronze.ipynb
25/12/15 19:44:39 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature canno
t be used because libhadoop cannot be loaded.
_
```



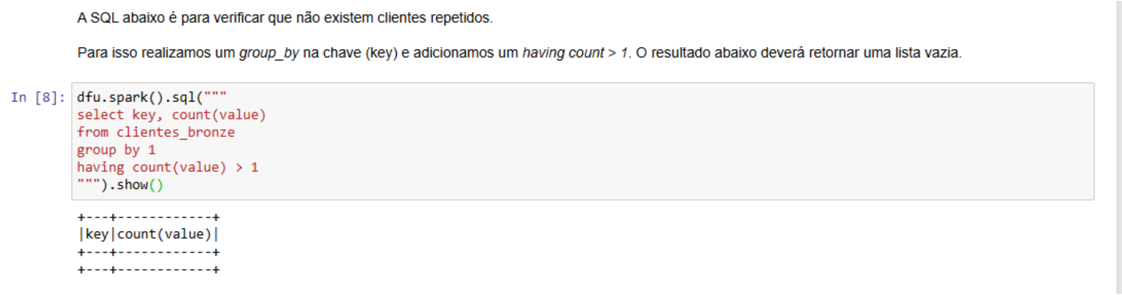
Veja que não é necessário recriar o Dataframe para que o valor seja atualizado, o Spark já cuida dessa parte para você. O valor final deverá ser **310.976**.

```
In [4]: pedidos.count()

Out[4]: 310976
```



A quantidade de clientes é muito pequena, portanto quando você executar o trecho abaixo já terá finalizado a escrita dos **15.000** clientes.

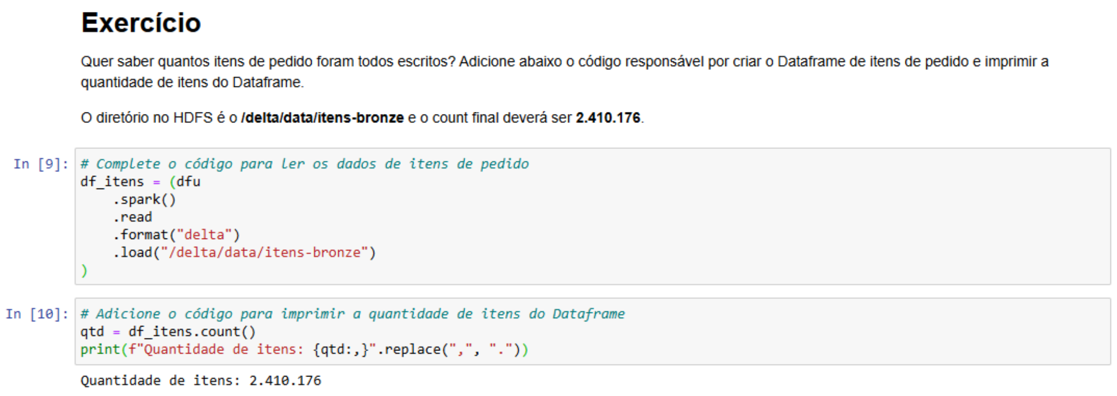
```
In [6]: clientes.count()

Out[6]: 15000
```



A SQL abaixo é para verificar que não existem clientes repetidos.

Para isso realizamos um *group_by* na chave (key) e adicionamos um *having count > 1*. O resultado abaixo deverá retornar uma lista vazia.

```
In [8]: dfu.spark().sql("""
        select key, count(value)
        from clientes_bronze
        group by 1
        having count(value) > 1
        """).show()

        +---+------------+
        |key|count(value)|
        +---+------------+
        +---+------------+
```



Exercício

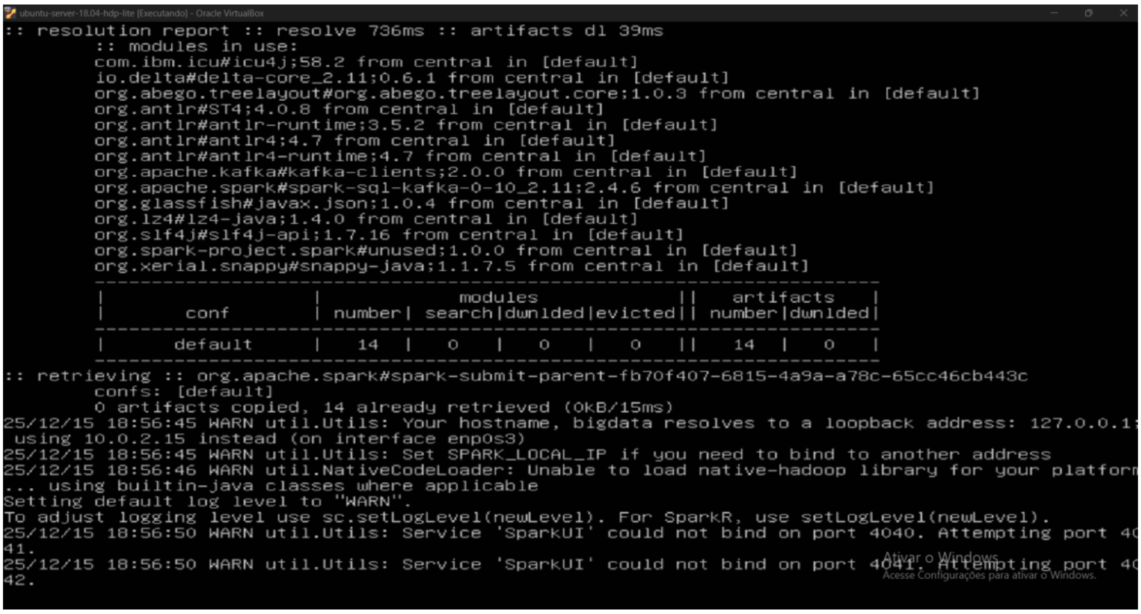
Quer saber quantos itens de pedido foram todos escritos? Adicione abaixo o código responsável por criar o Dataframe de itens de pedido e imprimir a quantidade de itens do Dataframe.

O diretório no HDFS é o **/delta/data/itens-bronze** e o count final deverá ser **2.410.176**.

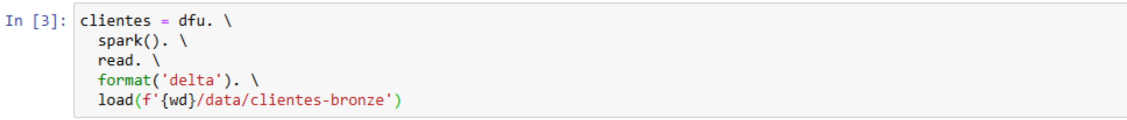
```
In [9]: # Complete o código para ler os dados de itens de pedido
        df_itens = (dfu
            .spark()
            .read
            .format("delta")
            .load("/delta/data/itens-bronze")
        )

In [10]: # Adicione o código para imprimir a quantidade de itens do Dataframe
         qtd = df_itens.count()
         print(f"Quantidade de itens: {qtd:,}".replace(",", "."))

         Quantidade de itens: 2.410.176
```

Rodando o 3-bronze2silver

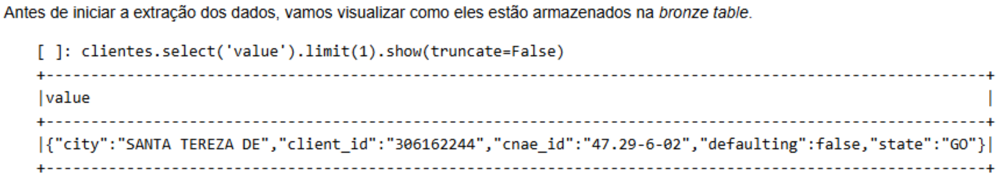
Arquivo *config.yaml* tem os dados de acesso ao Postgres e Kafka.


ubuntu-server-18.04-hdp-lite [Executando] - Oracle VirtualBox

```
:: resolution report :: resolve 736ms :: artifacts dl 39ms
        :: modules in use:
        com.ibm.icu#icu4j;58.2 from central in [default]
        io.delta#delta-core_2.11;0.6.1 from central in [default]
        org.abego.treelayout#org.abego.treelayout.core;1.0.3 from central in [default]
        org.antlr#ST4;4.0.8 from central in [default]
        org.antlr#antlr-runtime;3.5.2 from central in [default]
        org.antlr#antlr4;4.7 from central in [default]
        org.antlr#antlr4-runtime;4.7 from central in [default]
        org.apache.kafka#kafka-clients;2.0.0 from central in [default]
        org.apache.spark#spark-sql-kafka-0-10_2.11;2.4.6 from central in [default]
        org.glassfish#javax.json;1.0.4 from central in [default]
        org.lz4#lz4-java;1.4.0 from central in [default]
        org.slf4j#slf4j-api;1.7.16 from central in [default]
        org.spark-project.spark#unused;1.0.0 from central in [default]
        org.xerial.snappy#snappy-java;1.1.7.5 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   14  |   0   |   0   |   0   ||   14  |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-fb70f407-6815-4a9a-a78c-65cc46cb443c
        confs: [default]
        0 artifacts copied, 14 already retrieved (0kB/15ms)
25/12/15 18:56:45 WARN util.Utils: Your hostname, bigdata resolves to a loopback address: 127.0.0.1;
 using 10.0.2.15 instead (on interface enp0s3)
25/12/15 18:56:45 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another address
25/12/15 18:56:46 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platforn
... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/12/15 18:56:50 WARN util.Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 40
41.
25/12/15 18:56:50 WARN util.Utils: Service 'SparkUI' could not bind on port 4041. Attempting port 40
42.
```

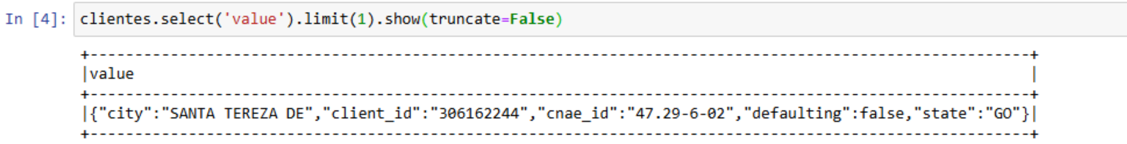


```
In [3]: clientes = dfu. \
          spark(). \
          read. \
          format('delta'). \
          load(f'{wd}/data/clientes-bronze')
```

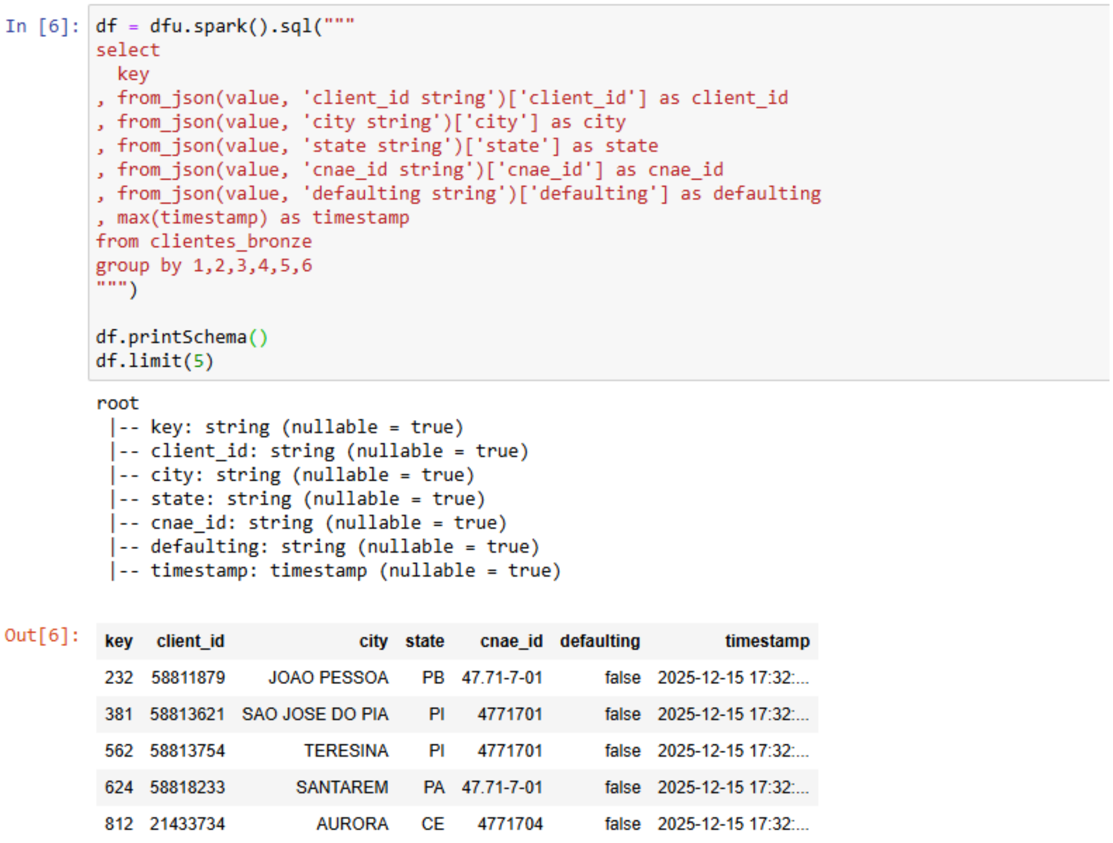


Antes de iniciar a extração dos dados, vamos visualizar como eles estão armazenados na *bronze table*.

```
[ ]: clientes.select('value').limit(1).show(truncate=False)

+-----------------------------------------------------------------------------------------------------+
|value                                                                                                |
+-----------------------------------------------------------------------------------------------------+
|{"city":"SANTA TEREZA DE","client_id":"306162244","cnae_id":"47.29-6-02","defaulting":false,"state":"GO"}|
+-----------------------------------------------------------------------------------------------------+
```



```
In [4]: clientes.select('value').limit(1).show(truncate=False)

        +-----------------------------------------------------------------------------------------------------+
        |value                                                                                                |
        +-----------------------------------------------------------------------------------------------------+
        |{"city":"SANTA TEREZA DE","client_id":"306162244","cnae_id":"47.29-6-02","defaulting":false,"state":"GO"}|
        +-----------------------------------------------------------------------------------------------------+
```

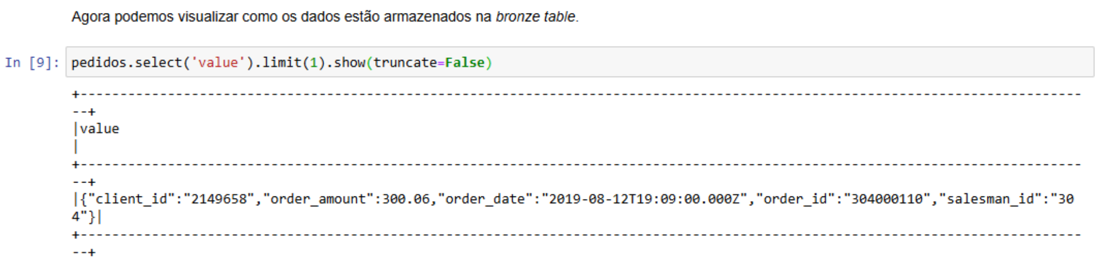


In [6]:
```
df = dfu.spark().sql("""
select
  key
, from_json(value, 'client_id string')['client_id'] as client_id
, from_json(value, 'city string')['city'] as city
, from_json(value, 'state string')['state'] as state
, from_json(value, 'cnae_id string')['cnae_id'] as cnae_id
, from_json(value, 'defaulting string')['defaulting'] as defaulting
, max(timestamp) as timestamp
from clientes_bronze
group by 1,2,3,4,5,6
""")

df.printSchema()
df.limit(5)
```

```
root
 |-- key: string (nullable = true)
 |-- client_id: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- cnae_id: string (nullable = true)
 |-- defaulting: string (nullable = true)
 |-- timestamp: timestamp (nullable = true)
```

Out[6]:

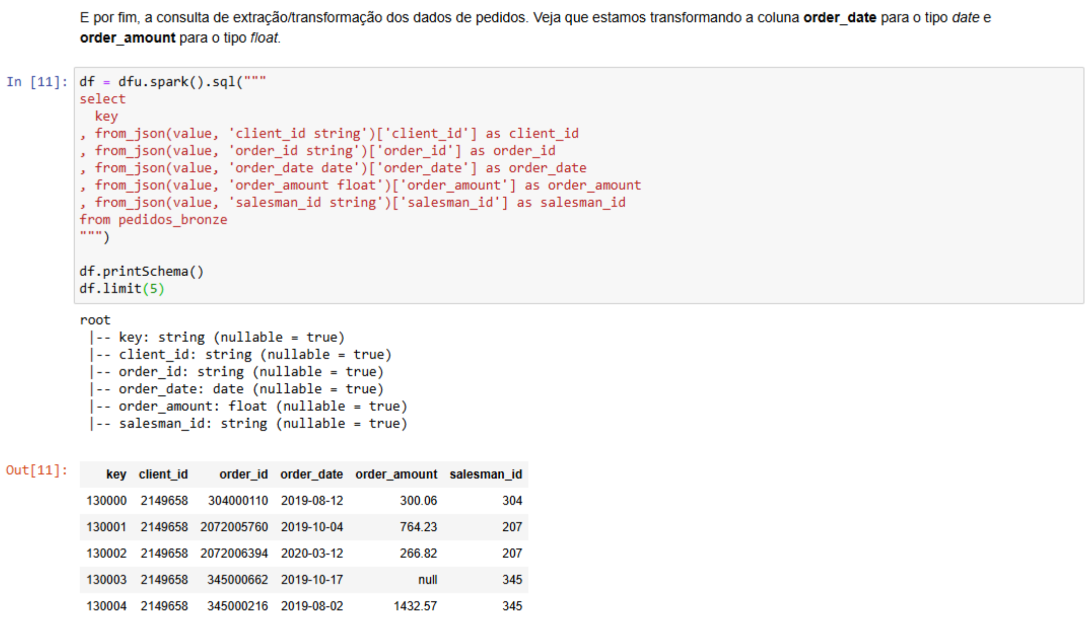
key	client_id	city	state	cnae_id	defaulting	timestamp
232	58811879	JOAO PESSOA	PB	47.71-7-01	false	2025-12-15 17:32:...
381	58813621	SAO JOSE DO PIA	PI	4771701	false	2025-12-15 17:32:...
562	58813754	TERESINA	PI	4771701	false	2025-12-15 17:32:...
624	58818233	SANTAREM	PA	47.71-7-01	false	2025-12-15 17:32:...
812	21433734	AURORA	CE	4771704	false	2025-12-15 17:32:...



Agora podemos visualizar como os dados estão armazenados na *bronze table*.

In [9]:
```
pedidos.select('value').limit(1).show(truncate=False)
```

```
+------------------------------------------------------------------------------------------------------------------------------
--+
|value
|
+------------------------------------------------------------------------------------------------------------------------------
--+
|{"client_id":"2149658","order_amount":300.06,"order_date":"2019-08-12T19:09:00.000Z","order_id":"304000110","salesman_id":"30
4"}|
+------------------------------------------------------------------------------------------------------------------------------
--+
```



E por fim, a consulta de extração/transformação dos dados de pedidos. Veja que estamos transformando a coluna **order_date** para o tipo *date* e **order_amount** para o tipo *float*.

In [11]:
```
df = dfu.spark().sql("""
select
  key
, from_json(value, 'client_id string')['client_id'] as client_id
, from_json(value, 'order_id string')['order_id'] as order_id
, from_json(value, 'order_date date')['order_date'] as order_date
, from_json(value, 'order_amount float')['order_amount'] as order_amount
, from_json(value, 'salesman_id string')['salesman_id'] as salesman_id
from pedidos_bronze
""")

df.printSchema()
df.limit(5)
```

```
root
 |-- key: string (nullable = true)
 |-- client_id: string (nullable = true)
 |-- order_id: string (nullable = true)
 |-- order_date: date (nullable = true)
 |-- order_amount: float (nullable = true)
 |-- salesman_id: string (nullable = true)
```

Out[11]:

key	client_id	order_id	order_date	order_amount	salesman_id
130000	2149658	304000110	2019-08-12	300.06	304
130001	2149658	2072005760	2019-10-04	764.23	207
130002	2149658	2072006394	2020-03-12	266.82	207
130003	2149658	345000662	2019-10-17	null	345
130004	2149658	345000216	2019-08-02	1432.57	345

Agora podemos visualizar como os dados estão armazenados na *bronze table*.

In [9]:
```
pedidos.select('value').limit(1).show(truncate=False)
```

```
+------------------------------------------------------------------------------------------------------------------------------
--+
|value
|
+------------------------------------------------------------------------------------------------------------------------------
--+
|{"client_id":"2149658","order_amount":300.06,"order_date":"2019-08-12T19:09:00.000Z","order_id":"304000110","salesman_id":"30
4"}|
+------------------------------------------------------------------------------------------------------------------------------
--+
```

E por fim, a consulta de extração/transformação dos dados de pedidos. Veja que estamos transformando a coluna **order_date** para o tipo *date* e **order_amount** para o tipo *float*.

In [11]:
```
df = dfu.spark().sql("""
select
  key
, from_json(value, 'client_id string')['client_id'] as client_id
, from_json(value, 'order_id string')['order_id'] as order_id
, from_json(value, 'order_date date')['order_date'] as order_date
, from_json(value, 'order_amount float')['order_amount'] as order_amount
, from_json(value, 'salesman_id string')['salesman_id'] as salesman_id
from pedidos_bronze
""")

df.printSchema()
df.limit(5)
```

```
root
 |-- key: string (nullable = true)
 |-- client_id: string (nullable = true)
 |-- order_id: string (nullable = true)
 |-- order_date: date (nullable = true)
 |-- order_amount: float (nullable = true)
 |-- salesman_id: string (nullable = true)
```

Out[11]:

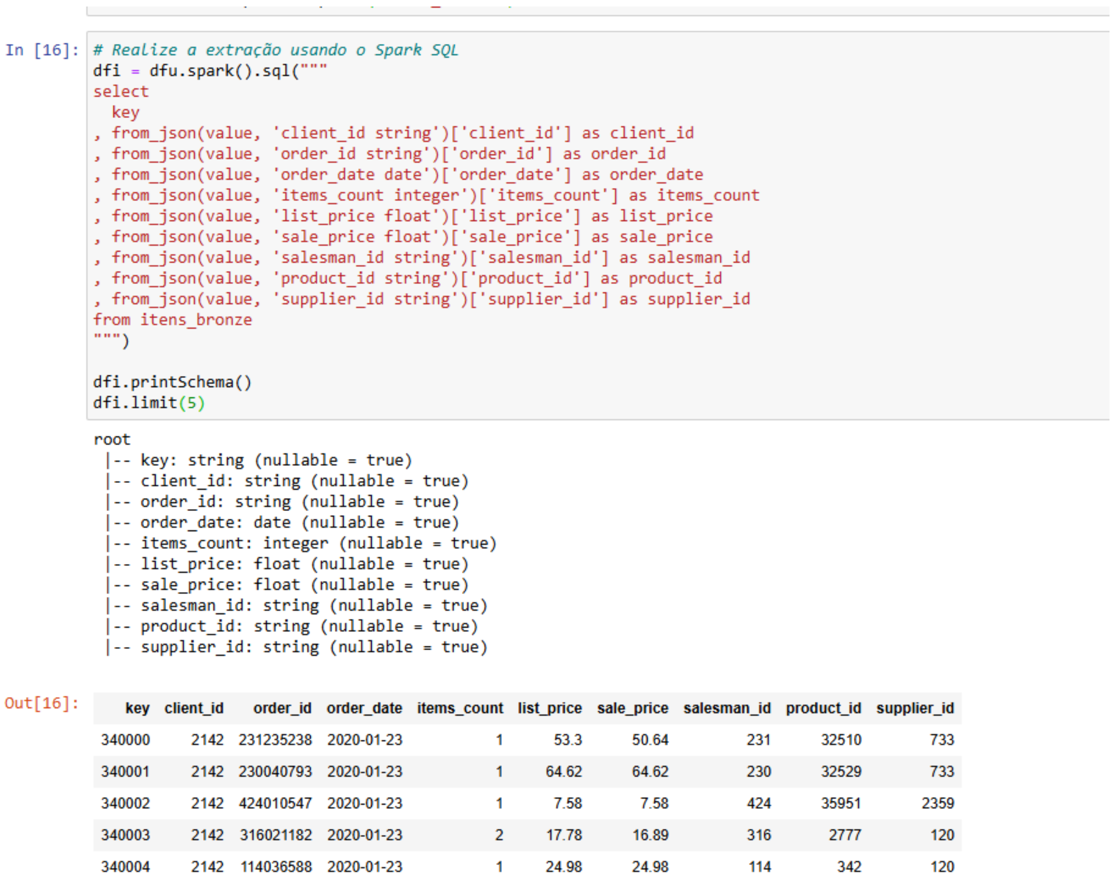
key	client_id	order_id	order_date	order_amount	salesman_id
130000	2149658	304000110	2019-08-12	300.06	304
130001	2149658	2072005760	2019-10-04	764.23	207
130002	2149658	2072006394	2020-03-12	266.82	207
130003	2149658	345000662	2019-10-17	null	345
130004	2149658	345000216	2019-08-02	1432.57	345

4. Escrever os novos dados em /delta/data/itens-silver.

In [13]:
```
# Comece criando o dataframe
itens = dfu. \
  spark(). \
  read. \
  format('delta'). \
  load(f'{wd}/data/itens-bronze')
```

In [14]:
```
# Veja como os dados estão armazenados
itens.select('value').limit(1).show(truncate=False)
```

```
+----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------+
|value
|
+----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------+
|{"client_id":"2142","items_count":1,"list_price":53.3,"order_date":"2020-01-23T03:00:00.000Z","order_id":"231235238","product_
id":"32510","sale_price":50.64,"salesman_id":"231","supplier_id":"733"}|
+----------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------+
```



In [16]:

```
# Realize a extração usando o Spark SQL
dfi = dfu.spark().sql("""
select
  key
, from_json(value, 'client_id string')['client_id'] as client_id
, from_json(value, 'order_id string')['order_id'] as order_id
, from_json(value, 'order_date date')['order_date'] as order_date
, from_json(value, 'items_count integer')['items_count'] as items_count
, from_json(value, 'list_price float')['list_price'] as list_price
, from_json(value, 'sale_price float')['sale_price'] as sale_price
, from_json(value, 'salesman_id string')['salesman_id'] as salesman_id
, from_json(value, 'product_id string')['product_id'] as product_id
, from_json(value, 'supplier_id string')['supplier_id'] as supplier_id
from itens_bronze
""")

dfi.printSchema()
dfi.limit(5)
```

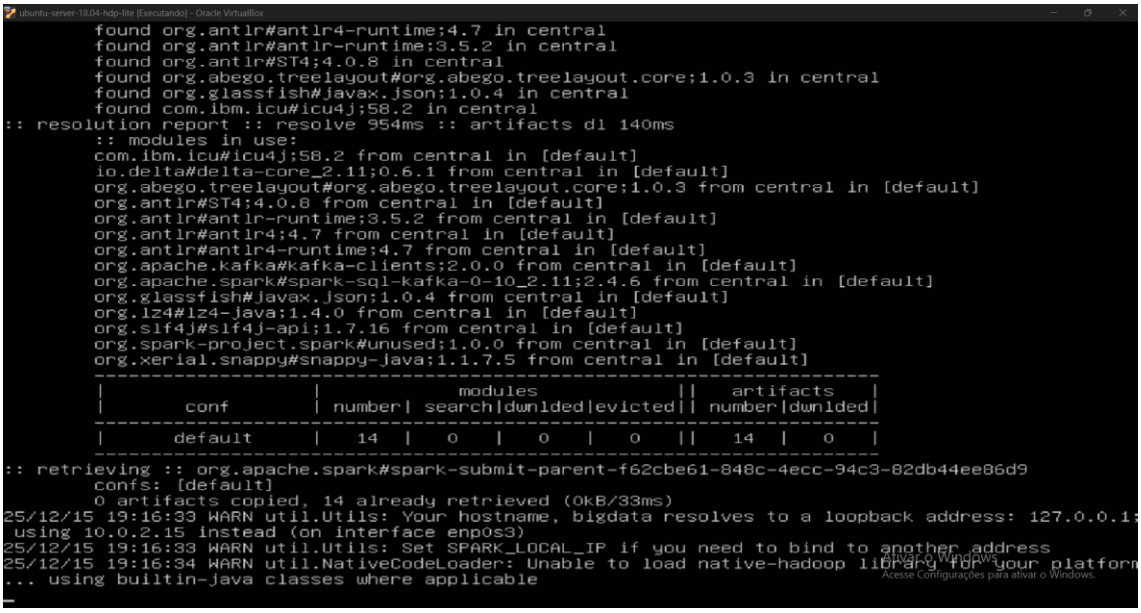
```
root
 |-- key: string (nullable = true)
 |-- client_id: string (nullable = true)
 |-- order_id: string (nullable = true)
 |-- order_date: date (nullable = true)
 |-- items_count: integer (nullable = true)
 |-- list_price: float (nullable = true)
 |-- sale_price: float (nullable = true)
 |-- salesman_id: string (nullable = true)
 |-- product_id: string (nullable = true)
 |-- supplier_id: string (nullable = true)
```

Out[16]:

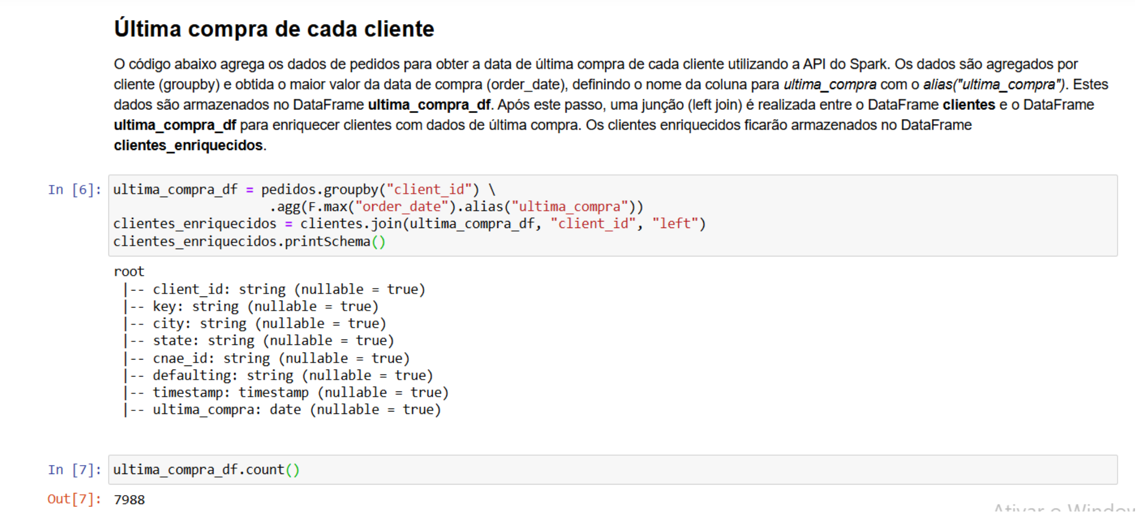
key	client_id	order_id	order_date	items_count	list_price	sale_price	salesman_id	product_id	supplier_id
340000	2142	231235238	2020-01-23	1	53.3	50.64	231	32510	733
340001	2142	230040793	2020-01-23	1	64.62	64.62	230	32529	733
340002	2142	424010547	2020-01-23	1	7.58	7.58	424	35951	2359
340003	2142	316021182	2020-01-23	2	17.78	16.89	316	2777	120
340004	2142	114036588	2020-01-23	1	24.98	24.98	114	342	120

Rodando 4-rfv

Arquivo *config.yaml* tem os dados de acesso ao Postgres e Kafka.


ubuntu-server-18.04-hdp-lite [Executando] - Oracle VirtualBox
Ativar o Windows
Acesse Configurações para ativar o Windows.

```
        found org.antlr#antlr4-runtime;4.7 in central
        found org.antlr#antlr-runtime;3.5.2 in central
        found org.antlr#ST4;4.0.8 in central
        found org.abego.treelayout#org.abego.treelayout.core;1.0.3 in central
        found org.glassfish#javax.json;1.0.4 in central
        found com.ibm.icu#icu4j;58.2 in central
:: resolution report :: resolve 954ms :: artifacts dl 140ms
        :: modules in use:
        com.ibm.icu#icu4j;58.2 from central in [default]
        io.delta#delta-core_2.11;0.6.1 from central in [default]
        org.abego.treelayout#org.abego.treelayout.core;1.0.3 from central in [default]
        org.antlr#ST4;4.0.8 from central in [default]
        org.antlr#antlr-runtime;3.5.2 from central in [default]
        org.antlr#antlr4;4.7 from central in [default]
        org.antlr#antlr4-runtime;4.7 from central in [default]
        org.apache.kafka#kafka-clients;2.0.0 from central in [default]
        org.apache.spark#spark-sql-kafka-0-10_2.11;2.4.6 from central in [default]
        org.glassfish#javax.json;1.0.4 from central in [default]
        org.lz4#lz4-java;1.4.0 from central in [default]
        org.slf4j#slf4j-api;1.7.16 from central in [default]
        org.spark-project.spark#unused;1.0.0 from central in [default]
        org.xerial.snappy#snappy-java;1.1.7.5 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   14  |   0   |   0   |   0   ||   14  |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-f62cbe61-848c-4ecc-94c3-82db44ee86d9
        confs: [default]
        0 artifacts copied, 14 already retrieved (0kB/33ms)
25/12/15 19:16:33 WARN util.Utils: Your hostname, bigdata resolves to a loopback address: 127.0.0.1;
 using 10.0.2.15 instead (on interface enp0s3)
25/12/15 19:16:33 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another address
25/12/15 19:16:34 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform
... using builtin-java classes where applicable
_
```



Última compra de cada cliente

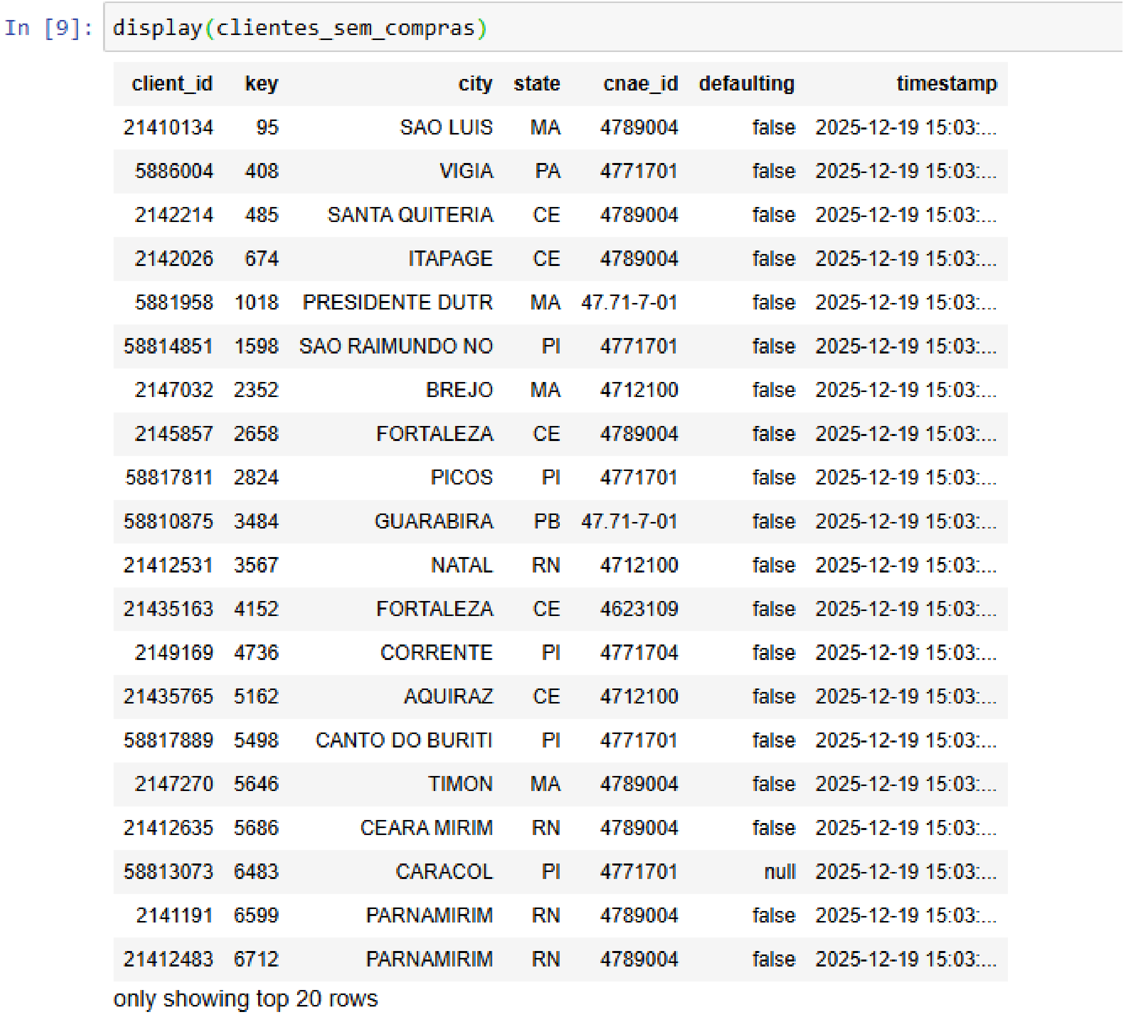
O código abaixo agrega os dados de pedidos para obter a data de última compra de cada cliente utilizando a API do Spark. Os dados são agregados por cliente (groupby) e obtida o maior valor da data de compra (order_date), definindo o nome da coluna para *ultima_compra* com o *alias("ultima_compra")*. Estes dados são armazenados no DataFrame **ultima_compra_df**. Após este passo, uma junção (left join) é realizada entre o DataFrame **clientes** e o DataFrame **ultima_compra_df** para enriquecer clientes com dados de última compra. Os clientes enriquecidos ficarão armazenados no DataFrame **clientes_enriquecidos**.

In [6]:
```
ultima_compra_df = pedidos.groupby("client_id") \
                    .agg(F.max("order_date").alias("ultima_compra"))
clientes_enriquecidos = clientes.join(ultima_compra_df, "client_id", "left")
clientes_enriquecidos.printSchema()
```

```
root
 |-- client_id: string (nullable = true)
 |-- key: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- cnae_id: string (nullable = true)
 |-- defaulting: string (nullable = true)
 |-- timestamp: timestamp (nullable = true)
 |-- ultima_compra: date (nullable = true)
```

In [7]:
```
ultima_compra_df.count()
```

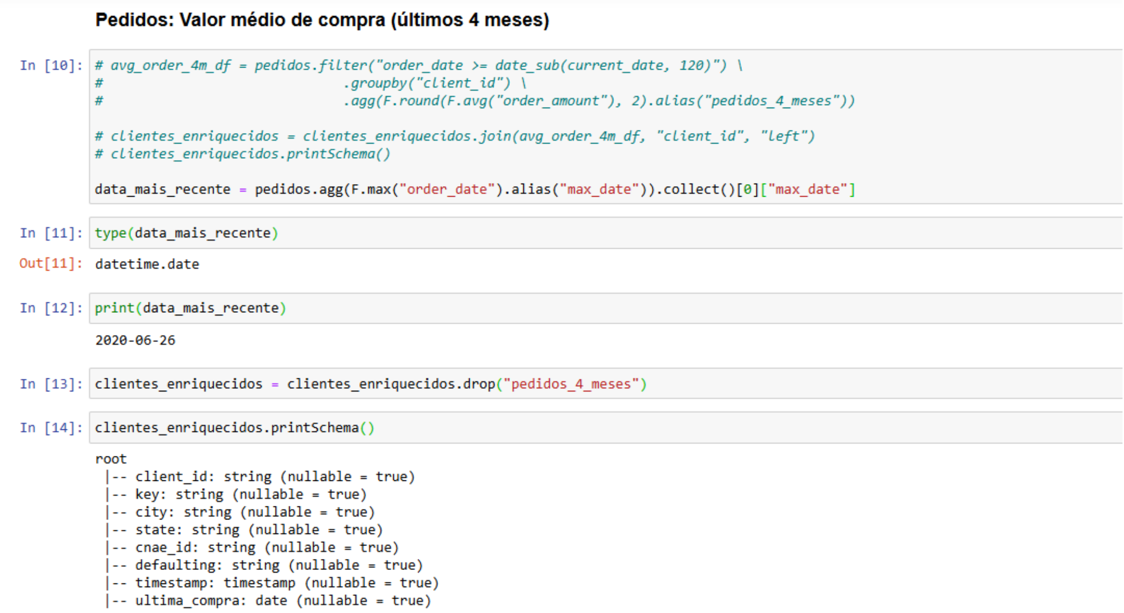
Out[7]: 7988



```
In [9]: display(clientes_sem_compras)
```

client_id	key	city	state	cnae_id	defaulting	timestamp
21410134	95	SAO LUIS	MA	4789004	false	2025-12-19 15:03:...
5886004	408	VIGIA	PA	4771701	false	2025-12-19 15:03:...
2142214	485	SANTA QUITERIA	CE	4789004	false	2025-12-19 15:03:...
2142026	674	ITAPAGE	CE	4789004	false	2025-12-19 15:03:...
5881958	1018	PRESIDENTE DUTR	MA	47.71-7-01	false	2025-12-19 15:03:...
58814851	1598	SAO RAIMUNDO NO	PI	4771701	false	2025-12-19 15:03:...
2147032	2352	BREJO	MA	4712100	false	2025-12-19 15:03:...
2145857	2658	FORTALEZA	CE	4789004	false	2025-12-19 15:03:...
58817811	2824	PICOS	PI	4771701	false	2025-12-19 15:03:...
58810875	3484	GUARABIRA	PB	47.71-7-01	false	2025-12-19 15:03:...
21412531	3567	NATAL	RN	4712100	false	2025-12-19 15:03:...
21435163	4152	FORTALEZA	CE	4623109	false	2025-12-19 15:03:...
2149169	4736	CORRENTE	PI	4771704	false	2025-12-19 15:03:...
21435765	5162	AQUIRAZ	CE	4712100	false	2025-12-19 15:03:...
58817889	5498	CANTO DO BURITI	PI	4771701	false	2025-12-19 15:03:...
2147270	5646	TIMON	MA	4789004	false	2025-12-19 15:03:...
21412635	5686	CEARA MIRIM	RN	4789004	false	2025-12-19 15:03:...
58813073	6483	CARACOL	PI	4771701	null	2025-12-19 15:03:...
2141191	6599	PARNAMIRIM	RN	4789004	false	2025-12-19 15:03:...
21412483	6712	PARNAMIRIM	RN	4789004	false	2025-12-19 15:03:...

only showing top 20 rows



Pedidos: Valor médio de compra (últimos 4 meses)

```
In [10]: # avg_order_4m_df = pedidos.filter("order_date >= date_sub(current_date, 120)") \
         #                               .groupby("client_id") \
         #                               .agg(F.round(F.avg("order_amount"), 2).alias("pedidos_4_meses"))

         # clientes_enriquecidos = clientes_enriquecidos.join(avg_order_4m_df, "client_id", "left")
         # clientes_enriquecidos.printSchema()

         data_mais_recente = pedidos.agg(F.max("order_date").alias("max_date")).collect()[0]["max_date"]

In [11]: type(data_mais_recente)

Out[11]: datetime.date

In [12]: print(data_mais_recente)

         2020-06-26

In [13]: clientes_enriquecidos = clientes_enriquecidos.drop("pedidos_4_meses")

In [14]: clientes_enriquecidos.printSchema()

         root
          |-- client_id: string (nullable = true)
          |-- key: string (nullable = true)
          |-- city: string (nullable = true)
          |-- state: string (nullable = true)
          |-- cnae_id: string (nullable = true)
          |-- defaulting: string (nullable = true)
          |-- timestamp: timestamp (nullable = true)
          |-- ultima_compra: date (nullable = true)
```

In [15]:

```
# Usar a linha abaixo caso os dados forem históricos ou simulados

avg_order_4m_df = pedidos \
    .filter(F.col("order_date") >= F.date_sub(F.lit(data_mais_recente), 120)) \
    .groupby("client_id") \
    .agg(F.round(F.avg("order_amount"), 2).alias("pedidos_4_meses"))

# Usar a linha abaixo caso os dados estejam sendo alimentados constantemente
# avg_order_4m_df = pedidos.filter("order_date >= date_sub(current_date, 120)") \
#                             .groupby("client_id") \
#                             .agg(F.round(F.avg("order_amount"), 2).alias("pedidos_4_meses"))

clientes_enriquecidos = clientes_enriquecidos.join(avg_order_4m_df, "client_id", "left")
clientes_enriquecidos.printSchema()
```

```
root
 |-- client_id: string (nullable = true)
 |-- key: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- cnae_id: string (nullable = true)
 |-- defaulting: string (nullable = true)
 |-- timestamp: timestamp (nullable = true)
 |-- ultima_compra: date (nullable = true)
 |-- pedidos_4_meses: double (nullable = true)
```

In [16]:

```
display(clientes_enriquecidos)
```

client_id	key	city	state	cnae_id	defaulting	timestamp	ultima_compra	pedidos_4_meses
21410134	95	SAO LUIS	MA	4789004	false	2025-12-19 15:03:...	null	null
5886004	408	VIGIA	PA	4771701	false	2025-12-19 15:03:...	null	null
5889897	412	JOAO PESSOA	PB	4771701	false	2025-12-19 15:03:...	2020-06-23	462.82
58822433	471	ESPERANTINA	PI	4771701	false	2025-12-19 15:03:...	2020-06-25	211.64
2142214	485	SANTA QUITERIA	CE	4789004	false	2025-12-19 15:03:...	null	null
2142026	674	ITAPAGE	CE	4789004	false	2025-12-19 15:03:...	null	null
21436937	743	PAU DOS FERROS	RN	4789004	false	2025-12-19 15:03:...	2020-06-08	771.82
21411791	961	PARNAMIRIM	RN	4789004	false	2025-12-19 15:03:...	2020-05-22	588.82
5881958	1018	PRESIDENTE DUTR	MA	47.71-7-01	false	2025-12-19 15:03:...	null	null
58814862	1307	ANGICAL DO PIAU	PI	4771701	false	2025-12-19 15:03:...	2020-06-09	707.98
2149478	1391	TERESINA	PI	8122200	false	2025-12-19 15:03:...	2020-06-19	344.02
21437159	1581	FARIAS BRITO	CE	8122200	false	2025-12-19 15:03:...	2020-02-04	null
58814851	1598	SAO RAIMUNDO NO	PI	4771701	false	2025-12-19 15:03:...	null	null
58814274	1718	CASTELO DO PIAU	PI	4771701	false	2025-12-19 15:03:...	2020-06-25	338.09
5886297	1743	ALTAMIRA	PA	47.71-7-01	false	2025-12-19 15:03:...	2020-06-19	613.83
21411956	1889	MOGEIRO	PB	0151201	false	2025-12-19 15:03:...	2019-10-22	null
58811830	2039	ITABAIANA	PB	4771-7/01	false	2025-12-19 15:03:...	2020-06-23	497.72
2147032	2352	BREJO	MA	4712100	false	2025-12-19 15:03:...	null	null
58822141	2407	BELEM	PA	G4771701	false	2025-12-19 15:03:...	2020-03-25	774.14
2145857	2658	FORTALEZA	CE	4789004	false	2025-12-19 15:03:...	null	null

only showing top 20 rows

In [17]:

```
clientes_enriquecidos.filter(F.col("pedidos_4_meses").isNotNull()).count()
```

Out[17]:

```
6541
```

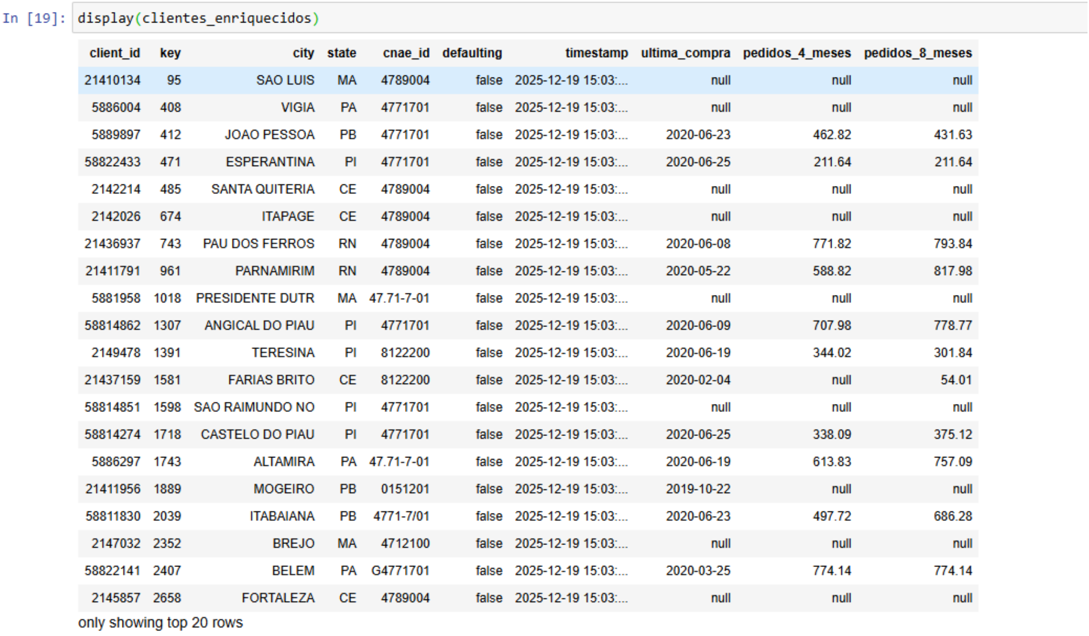
Exercício - Pedidos: Valor médio de compra (últimos 8 meses)

Altere o filtro de dados e realizar a consulta para calcular o valor médio de pedidos dos últimos 8 meses. Enriqueça o DataFrame **clientes_enriquecidos** com uma nova coluna **pedidos_8_meses** com o valor médio de compra dos últimos 8 meses.

In [18]:

```
# Usar a linha abaixo caso os dados forem históricos ou simulados

avg_order_8m_df = pedidos \
    .filter(F.col("order_date") >= F.date_sub(F.lit(data_mais_recente), 240)) \
    .groupby("client_id") \
    .agg(F.round(F.avg("order_amount"), 2).alias("pedidos_8_meses"))

# Usar a linha abaixo caso os dados estejam sendo alimentados constantemente
# avg_order_4m_df = pedidos.filter("order_date >= date_sub(current_date, 120)") \
#                             .groupby("client_id") \
#                             .agg(F.round(F.avg("order_amount"), 2).alias("pedidos_4_meses"))

clientes_enriquecidos = clientes_enriquecidos.join(avg_order_8m_df, "client_id", "left")
clientes_enriquecidos.printSchema()
```

```
root
 |-- client_id: string (nullable = true)
 |-- key: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- cnae_id: string (nullable = true)
 |-- defaulting: string (nullable = true)
 |-- timestamp: timestamp (nullable = true)
 |-- ultima_compra: date (nullable = true)
 |-- pedidos_4_meses: double (nullable = true)
 |-- pedidos_8_meses: double (nullable = true)
```



```
In [19]: display(clientes_enriquecidos)
```

client_id	key	city	state	cnae_id	defaulting	timestamp	ultima_compra	pedidos_4_meses	pedidos_8_meses
21410134	95	SAO LUIS	MA	4789004	false	2025-12-19 15:03:...	null	null	null
5886004	408	VIGIA	PA	4771701	false	2025-12-19 15:03:...	null	null	null
5889897	412	JOAO PESSOA	PB	4771701	false	2025-12-19 15:03:...	2020-06-23	462.82	431.63
58822433	471	ESPERANTINA	PI	4771701	false	2025-12-19 15:03:...	2020-06-25	211.64	211.64
2142214	485	SANTA QUITERIA	CE	4789004	false	2025-12-19 15:03:...	null	null	null
2142026	674	ITAPAGE	CE	4789004	false	2025-12-19 15:03:...	null	null	null
21436937	743	PAU DOS FERROS	RN	4789004	false	2025-12-19 15:03:...	2020-06-08	771.82	793.84
21411791	961	PARNAMIRIM	RN	4789004	false	2025-12-19 15:03:...	2020-05-22	588.82	817.98
5881958	1018	PRESIDENTE DUTR	MA	47.71-7-01	false	2025-12-19 15:03:...	null	null	null
58814862	1307	ANGICAL DO PIAU	PI	4771701	false	2025-12-19 15:03:...	2020-06-09	707.98	778.77
2149478	1391	TERESINA	PI	8122200	false	2025-12-19 15:03:...	2020-06-19	344.02	301.84
21437159	1581	FARIAS BRITO	CE	8122200	false	2025-12-19 15:03:...	2020-02-04	null	54.01
58814851	1598	SAO RAIMUNDO NO	PI	4771701	false	2025-12-19 15:03:...	null	null	null
58814274	1718	CASTELO DO PIAU	PI	4771701	false	2025-12-19 15:03:...	2020-06-25	338.09	375.12
5886297	1743	ALTAMIRA	PA	47.71-7-01	false	2025-12-19 15:03:...	2020-06-19	613.83	757.09
21411956	1889	MOGEIRO	PB	0151201	false	2025-12-19 15:03:...	2019-10-22	null	null
58811830	2039	ITABAIANA	PB	4771-7/01	false	2025-12-19 15:03:...	2020-06-23	497.72	686.28
2147032	2352	BREJO	MA	4712100	false	2025-12-19 15:03:...	null	null	null
58822141	2407	BELEM	PA	G4771701	false	2025-12-19 15:03:...	2020-03-25	774.14	774.14
2145857	2658	FORTALEZA	CE	4789004	false	2025-12-19 15:03:...	null	null	null

only showing top 20 rows

Exercício - Pedidos: Valor médio de compra (últimos 12 meses)

Altere o filtro de dados e realizar a consulta para calcular o valor médio de pedidos dos últimos 12 meses. Enriqueça o DataFrame **clientes_enriquecidos** com uma nova coluna **pedidos_12_meses** com o valor médio de compra dos últimos 12 meses.

In [20]:

```
# Usar a linha abaixo caso os dados forem históricos ou simulados

avg_order_12m_df = pedidos \
    .filter(F.col("order_date") >= F.date_sub(F.lit(data_mais_recente), 360)) \
    .groupby("client_id") \
    .agg(F.round(F.avg("order_amount"), 2).alias("pedidos_12_meses"))

# Usar a linha abaixo caso os dados estejam sendo alimentados constantemente
# avg_order_4m_df = pedidos.filter("order_date >= date_sub(current_date, 120)") \
#                              .groupby("client_id") \
#                              .agg(F.round(F.avg("order_amount"), 2).alias("pedidos_4_meses"))

clientes_enriquecidos = clientes_enriquecidos.join(avg_order_12m_df, "client_id", "left")
clientes_enriquecidos.printSchema()
```

```
root
 |-- client_id: string (nullable = true)
 |-- key: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- cnae_id: string (nullable = true)
 |-- defaulting: string (nullable = true)
 |-- timestamp: timestamp (nullable = true)
 |-- ultima_compra: date (nullable = true)
 |-- pedidos_4_meses: double (nullable = true)
 |-- pedidos_8_meses: double (nullable = true)
 |-- pedidos_12_meses: double (nullable = true)
```

In [21]:

```
display(clientes_enriquecidos)
```

client_id	key	city	state	cnae_id	defaulting	timestamp	ultima_compra	pedidos_4_meses	pedidos_8_meses	pedidos_12_meses
21410134	95	SAO LUIS	MA	4789004	false	2025-12-19 15:03:...	null	null	null	null
5886004	408	VIGIA	PA	4771701	false	2025-12-19 15:03:...	null	null	null	null
5889897	412	JOAO PESSOA	PB	4771701	false	2025-12-19 15:03:...	2020-06-23	462.82	431.63	431.63
58822433	471	ESPERANTINA	PI	4771701	false	2025-12-19 15:03:...	2020-06-25	211.64	211.64	211.64
2142214	485	SANTA QUITERIA	CE	4789004	false	2025-12-19 15:03:...	null	null	null	null
2142026	674	ITAPAGE	CE	4789004	false	2025-12-19 15:03:...	null	null	null	null
21436937	743	PAU DOS FERROS	RN	4789004	false	2025-12-19 15:03:...	2020-06-08	771.82	793.84	918.1
21411791	961	PARNAMIRIM	RN	4789004	false	2025-12-19 15:03:...	2020-05-22	588.82	817.98	1016.67
5881958	1018	PRESIDENTE DUTR	MA	47.71-7-01	false	2025-12-19 15:03:...	null	null	null	null
58814862	1307	ANGICAL DO PIAU	PI	4771701	false	2025-12-19 15:03:...	2020-06-09	707.98	778.77	765.14
2149478	1391	TERESINA	PI	8122200	false	2025-12-19 15:03:...	2020-06-19	344.02	301.84	329.76
21437159	1581	FARIAS BRITO	CE	8122200	false	2025-12-19 15:03:...	2020-02-04	null	54.01	44.11
58814851	1598	SAO RAIMUNDO NO	PI	4771701	false	2025-12-19 15:03:...	null	null	null	null
58814274	1718	CASTELO DO PIAU	PI	4771701	false	2025-12-19 15:03:...	2020-06-25	338.09	375.12	346.66
5886297	1743	ALTAMIRA	PA	47.71-7-01	false	2025-12-19 15:03:...	2020-06-19	613.83	757.09	836.48
21411956	1889	MOGEIRO	PB	0151201	false	2025-12-19 15:03:...	2019-10-22	null	null	1317.18
58811830	2039	ITABAIANA	PB	4771-7/01	false	2025-12-19 15:03:...	2020-06-23	497.72	686.28	692.51
2147032	2352	BREJO	MA	4712100	false	2025-12-19 15:03:...	null	null	null	null
58822141	2407	BELEM	PA	G4771701	false	2025-12-19 15:03:...	2020-03-25	774.14	774.14	774.14
2145857	2658	FORTALEZA	CE	4789004	false	2025-12-19 15:03:...	null	null	null	null

only showing top 20 rows

Exercício - Quantidade média de itens pedidos

Calcule o preço médio dos itens de pedidos dos últimos 4, 8 e 12 meses. Veja o código feito para pedidos e tente realizar o mesmo para itens de pedidos.

Dica: a coluna **items_count** representa o número de itens vendidos em cada compra.

In [22]:
```
display(itens)
```

key	client_id	order_id	order_date	items_count	list_price	sale_price	salesman_id	product_id	supplier_id
340000	2142	231235238	2020-01-23	1	53.3	50.64	231	32510	733
340001	2142	230040793	2020-01-23	1	64.62	64.62	230	32529	733
340002	2142	424010547	2020-01-23	1	7.58	7.58	424	35951	2359
340003	2142	316021182	2020-01-23	2	17.78	16.89	316	2777	120
340004	2142	114036588	2020-01-23	1	24.98	24.98	114	342	120
340005	2142	316021155	2020-01-23	1	51.11	48.6	316	472	1448
340006	2142	234035329	2020-01-23	1	7.09	7.09	234	31508	634
340007	2142	319022814	2020-01-23	1	60.6	60.6	319	16881	1056
340008	2142	2270047686	2020-01-23	1	13.7	13.7	227	8530	101
340009	2142	316021187	2020-01-23	1	13.7	13.1	316	8530	101
340010	2142	460002198	2020-01-23	2	13.4	12.75	460	18951	101
340011	2142	319022827	2020-01-23	1	33.96	33.96	319	5124	103
340012	2142	234035337	2020-01-23	1	10.99	10.99	234	10448	577
340013	2142	230040796	2020-01-23	1	5.34	5.34	230	37811	634
340014	2142	234035334	2020-01-23	1	3.9	3.9	234	28283	634
340015	2142	230040788	2020-01-23	1	18.11	17.2	230	37198	873
340016	2142	234035346	2020-01-23	3	81.15	77.09	234	36382	960
340017	2142	319022841	2020-01-23	1	163.92	163.92	319	25070	294
340018	2142	2270047698	2020-01-23	1	103.46	103.46	227	13656	336
340019	2142	231235215	2020-01-23	1	47.07	47.07	231	30766	336

only showing top 20 rows

Exercício - Quantidade média de itens de pedidos (4 meses)

Enriqueça o DataFrame **clientes_enriquecidos** com uma nova coluna **itens_4_meses** com a quantidade média de itens de pedidos vendidos nos últimos 4 meses.

In [23]:
```
avg_itens_4m_df = itens \
    .filter(F.col("order_date") >= F.date_sub(F.lit(data_mais_recente), 120)) \
    .groupby("client_id") \
    .agg(F.round(F.avg("items_count"), 2).alias("itens_4_meses"))


clientes_enriquecidos = clientes_enriquecidos.join(avg_itens_4m_df, "client_id", "left")
clientes_enriquecidos.printSchema()
```

```
root
 |-- client_id: string (nullable = true)
 |-- key: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- cnae_id: string (nullable = true)
 |-- defaulting: string (nullable = true)
 |-- timestamp: timestamp (nullable = true)
 |-- ultima_compra: date (nullable = true)
 |-- pedidos_4_meses: double (nullable = true)
 |-- pedidos_8_meses: double (nullable = true)
 |-- pedidos_12_meses: double (nullable = true)
 |-- itens_4_meses: double (nullable = true)
```

In [24]:

```
display(clientes_enriquecidos)
```

client_id	key	city	state	cnae_id	defaulting	timestamp	ultima_compra	pedidos_4_meses	pedidos_8_meses	pedidos_12_meses	itens_4_meses
21410134	95	SAO LUIS	MA	4789004	false	2025-12-19 15:03:...	null	null	null	null	nul
5886004	408	VIGIA	PA	4771701	false	2025-12-19 15:03:...	null	null	null	null	nul
5889897	412	JOAO PESSOA	PB	4771701	false	2025-12-19 15:03:...	2020-06-23	462.82	431.63	431.63	11.94
58822433	471	ESPERANTINA	PI	4771701	false	2025-12-19 15:03:...	2020-06-25	211.64	211.64	211.64	9.79
2142214	485	SANTA QUITERIA	CE	4789004	false	2025-12-19 15:03:...	null	null	null	null	nul
2142026	674	ITAPAGE	CE	4789004	false	2025-12-19 15:03:...	null	null	null	null	nul
21436937	743	PAU DOS FERROS	RN	4789004	false	2025-12-19 15:03:...	2020-06-08	771.82	793.84	918.1	4.84
21411791	961	PARNAMIRIM	RN	4789004	false	2025-12-19 15:03:...	2020-05-22	588.82	817.98	1016.67	5.06
5881958	1018	PRESIDENTE DUTR	MA	47.71-7-01	false	2025-12-19 15:03:...	null	null	null	null	nul
58814862	1307	ANGICAL DO PIAU	PI	4771701	false	2025-12-19 15:03:...	2020-06-09	707.98	778.77	765.14	41.95
2149478	1391	TERESINA	PI	8122200	false	2025-12-19 15:03:...	2020-06-19	344.02	301.84	329.76	2.6
21437159	1581	FARIAS BRITO	CE	8122200	false	2025-12-19 15:03:...	2020-02-04	null	54.01	44.11	nul
58814851	1598	SAO RAIMUNDO NO	PI	4771701	false	2025-12-19 15:03:...	null	null	null	null	nul
58814274	1718	CASTELO DO PIAU	PI	4771701	false	2025-12-19 15:03:...	2020-06-25	338.09	375.12	346.66	15.32
5886297	1743	ALTAMIRA	PA	47.71-7-01	false	2025-12-19 15:03:...	2020-06-19	613.83	757.09	836.48	18.07
21411956	1889	MOGEIRO	PB	0151201	false	2025-12-19 15:03:...	2019-10-22	null	null	1317.18	nul
58811830	2039	ITABAIANA	PB	4771-7/01	false	2025-12-19 15:03:...	2020-06-23	497.72	686.28	692.51	11.76

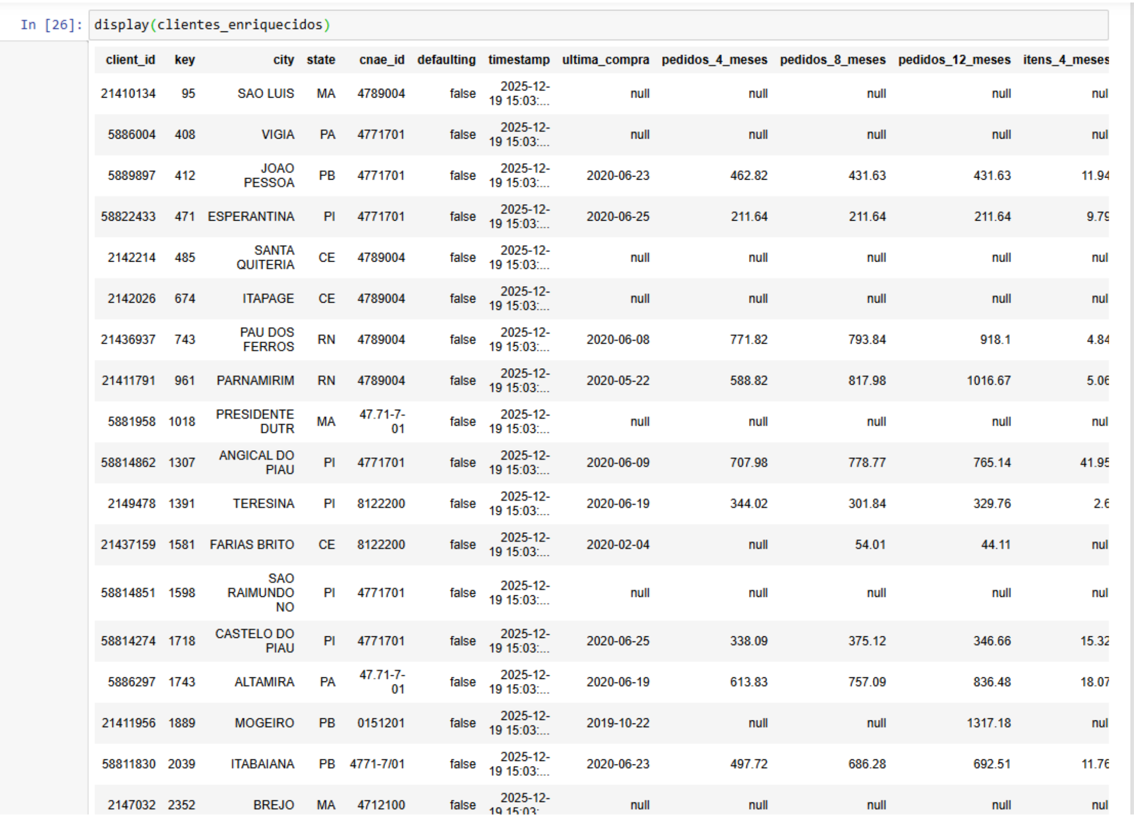
Exercício - Quantidade média de itens de pedidos (8 meses)

Enriqueça o DataFrame **clientes_enriquecidos** com uma nova coluna **itens_8_meses** com a quantidade média de itens de pedidos vendidos nos últimos 8 meses.

In [25]:

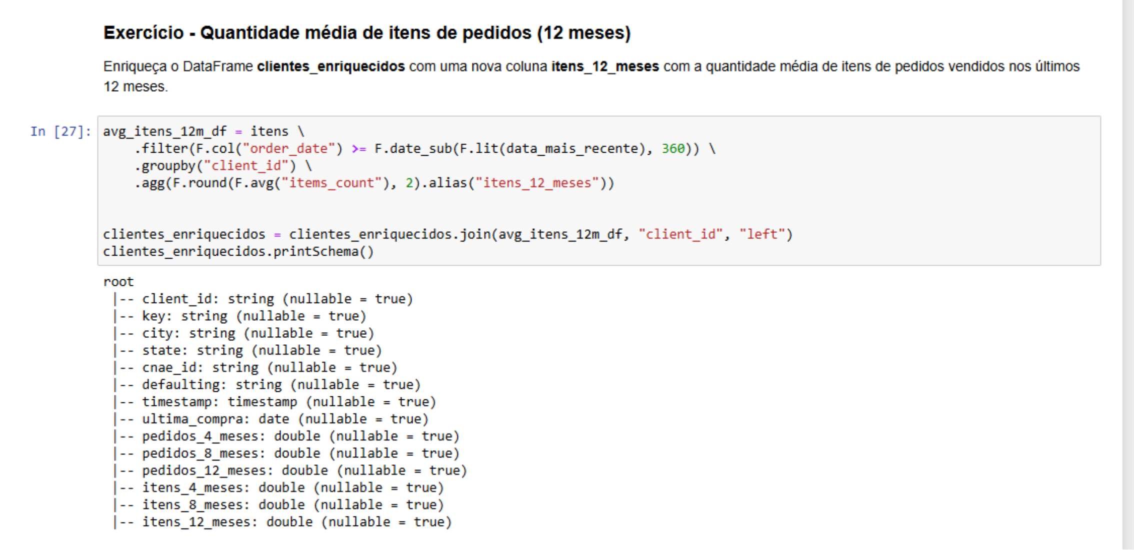
```
avg_itens_8m_df = itens \
    .filter(F.col("order_date") >= F.date_sub(F.lit(data_mais_recente), 240)) \
    .groupby("client_id") \
    .agg(F.round(F.avg("items_count"), 2).alias("itens_8_meses"))


clientes_enriquecidos = clientes_enriquecidos.join(avg_itens_8m_df, "client_id", "left")
clientes_enriquecidos.printSchema()
```

```
root
 |-- client_id: string (nullable = true)
 |-- key: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- cnae_id: string (nullable = true)
 |-- defaulting: string (nullable = true)
 |-- timestamp: timestamp (nullable = true)
 |-- ultima_compra: date (nullable = true)
 |-- pedidos_4_meses: double (nullable = true)
 |-- pedidos_8_meses: double (nullable = true)
 |-- pedidos_12_meses: double (nullable = true)
 |-- itens_4_meses: double (nullable = true)
 |-- itens_8_meses: double (nullable = true)
```



In [26]:
```
display(clientes_enriquecidos)
```

client_id	key	city	state	cnae_id	defaulting	timestamp	ultima_compra	pedidos_4_meses	pedidos_8_meses	pedidos_12_meses	itens_4_meses
21410134	95	SAO LUIS	MA	4789004	false	2025-12-19 15:03:...	null	null	null	null	nul
5886004	408	VIGIA	PA	4771701	false	2025-12-19 15:03:...	null	null	null	null	nul
5889897	412	JOAO PESSOA	PB	4771701	false	2025-12-19 15:03:...	2020-06-23	462.82	431.63	431.63	11.94
58822433	471	ESPERANTINA	PI	4771701	false	2025-12-19 15:03:...	2020-06-25	211.64	211.64	211.64	9.79
2142214	485	SANTA QUITERIA	CE	4789004	false	2025-12-19 15:03:...	null	null	null	null	nul
2142026	674	ITAPAGE	CE	4789004	false	2025-12-19 15:03:...	null	null	null	null	nul
21436937	743	PAU DOS FERROS	RN	4789004	false	2025-12-19 15:03:...	2020-06-08	771.82	793.84	918.1	4.84
21411791	961	PARNAMIRIM	RN	4789004	false	2025-12-19 15:03:...	2020-05-22	588.82	817.98	1016.67	5.06
5881958	1018	PRESIDENTE DUTR	MA	47.71-7-01	false	2025-12-19 15:03:...	null	null	null	null	nul
58814862	1307	ANGICAL DO PIAU	PI	4771701	false	2025-12-19 15:03:...	2020-06-09	707.98	778.77	765.14	41.95
2149478	1391	TERESINA	PI	8122200	false	2025-12-19 15:03:...	2020-06-19	344.02	301.84	329.76	2.6
21437159	1581	FARIAS BRITO	CE	8122200	false	2025-12-19 15:03:...	2020-02-04	null	54.01	44.11	nul
58814851	1598	SAO RAIMUNDO NO	PI	4771701	false	2025-12-19 15:03:...	null	null	null	null	nul
58814274	1718	CASTELO DO PIAU	PI	4771701	false	2025-12-19 15:03:...	2020-06-25	338.09	375.12	346.66	15.32
5886297	1743	ALTAMIRA	PA	47.71-7-01	false	2025-12-19 15:03:...	2020-06-19	613.83	757.09	836.48	18.07
21411956	1889	MOGEIRO	PB	0151201	false	2025-12-19 15:03:...	2019-10-22	null	null	1317.18	nul
58811830	2039	ITABAIANA	PB	4771-7/01	false	2025-12-19 15:03:...	2020-06-23	497.72	686.28	692.51	11.76
2147032	2352	BREJO	MA	4712100	false	2025-12-19 15:03:	null	null	null	null	nul

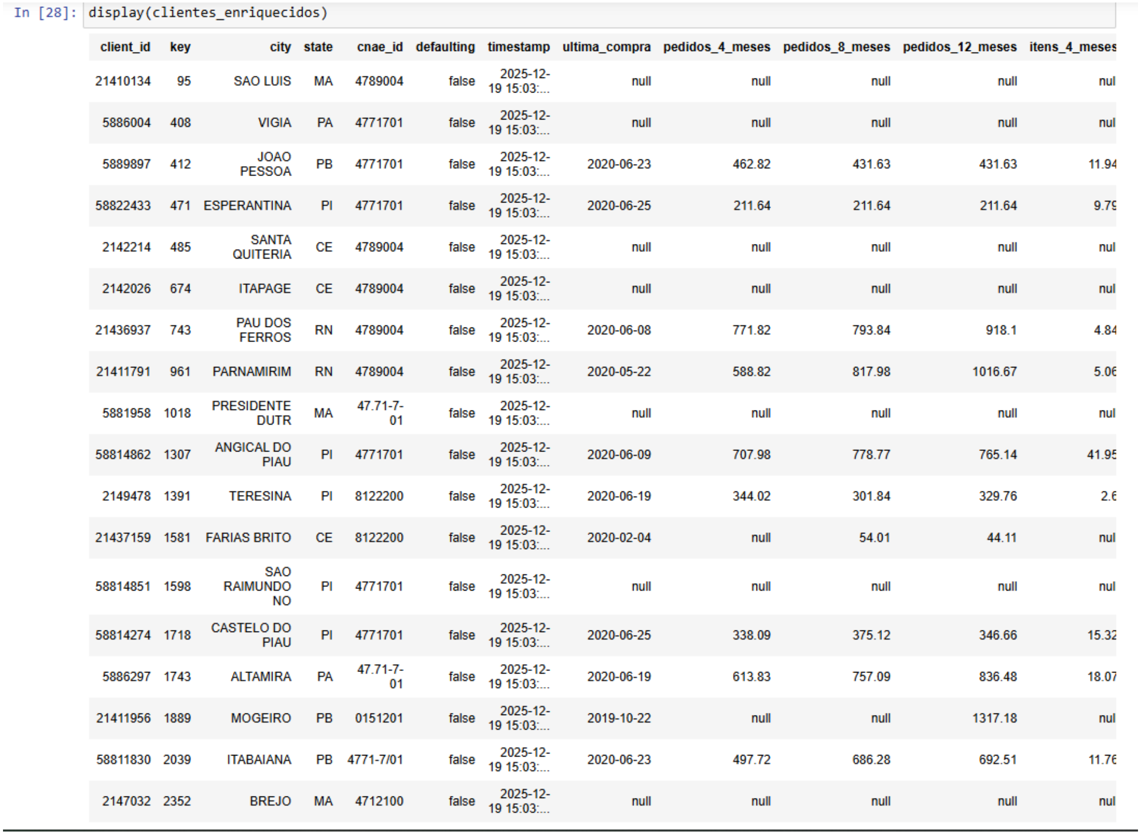


Exercício - Quantidade média de itens de pedidos (12 meses)

Enriqueça o DataFrame **clientes_enriquecidos** com uma nova coluna **itens_12_meses** com a quantidade média de itens de pedidos vendidos nos últimos 12 meses.

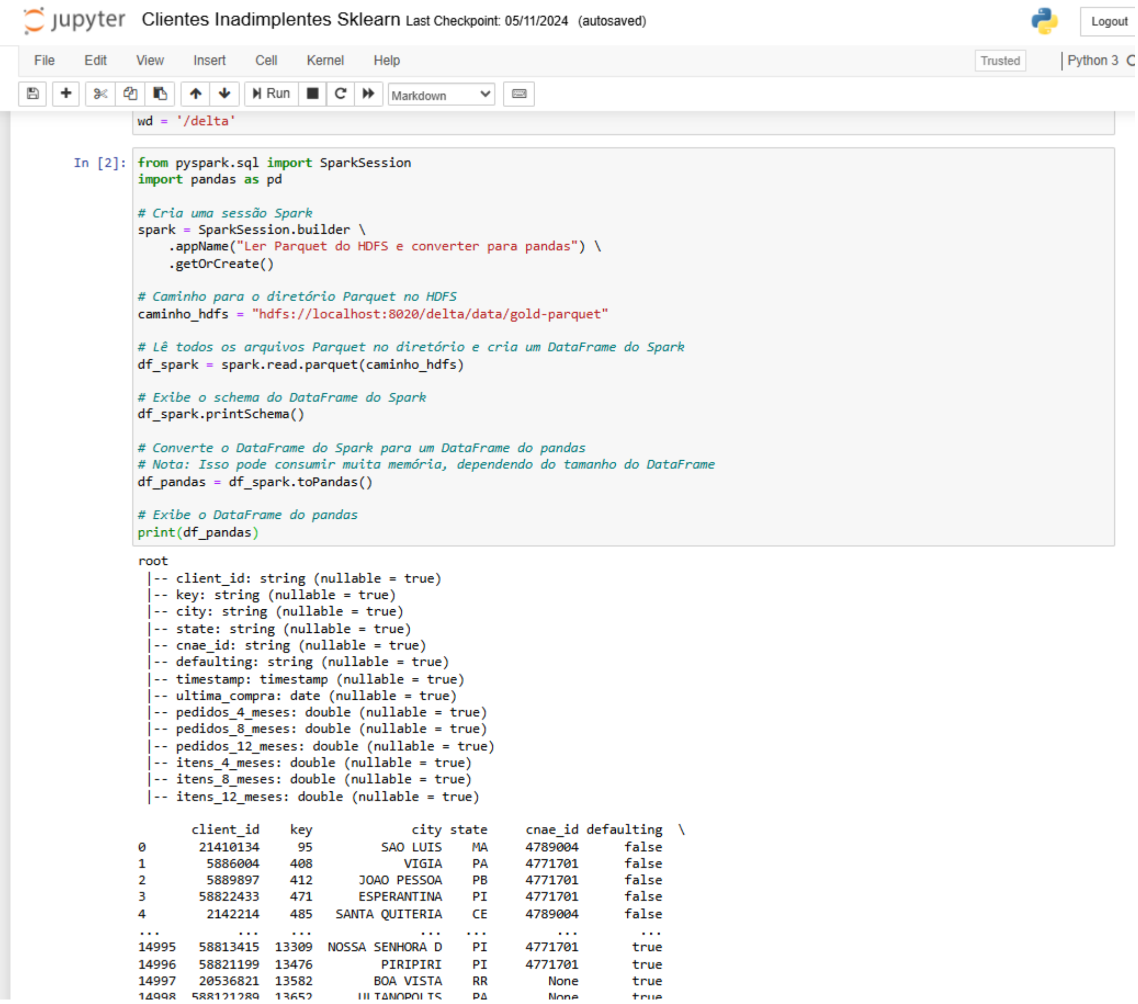
In [27]:
```
avg_itens_12m_df = itens \
    .filter(F.col("order_date") >= F.date_sub(F.lit(data_mais_recente), 360)) \
    .groupby("client_id") \
    .agg(F.round(F.avg("items_count"), 2).alias("itens_12_meses"))


clientes_enriquecidos = clientes_enriquecidos.join(avg_itens_12m_df, "client_id", "left")
clientes_enriquecidos.printSchema()
```

```
root
 |-- client_id: string (nullable = true)
 |-- key: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- cnae_id: string (nullable = true)
 |-- defaulting: string (nullable = true)
 |-- timestamp: timestamp (nullable = true)
 |-- ultima_compra: date (nullable = true)
 |-- pedidos_4_meses: double (nullable = true)
 |-- pedidos_8_meses: double (nullable = true)
 |-- pedidos_12_meses: double (nullable = true)
 |-- itens_4_meses: double (nullable = true)
 |-- itens_8_meses: double (nullable = true)
 |-- itens_12_meses: double (nullable = true)
```



```
In [28]: display(clientes_enriquecidos)
```

client_id	key	city	state	cnae_id	defaulting	timestamp	ultima_compra	pedidos_4_meses	pedidos_8_meses	pedidos_12_meses	itens_4_meses
21410134	95	SAO LUIS	MA	4789004	false	2025-12-19 15:03:...	null	null	null	null	nul
5886004	408	VIGIA	PA	4771701	false	2025-12-19 15:03:...	null	null	null	null	nul
5889897	412	JOAO PESSOA	PB	4771701	false	2025-12-19 15:03:...	2020-06-23	462.82	431.63	431.63	11.94
58822433	471	ESPERANTINA	PI	4771701	false	2025-12-19 15:03:...	2020-06-25	211.64	211.64	211.64	9.79
2142214	485	SANTA QUITERIA	CE	4789004	false	2025-12-19 15:03:...	null	null	null	null	nul
2142026	674	ITAPAGE	CE	4789004	false	2025-12-19 15:03:...	null	null	null	null	nul
21436937	743	PAU DOS FERROS	RN	4789004	false	2025-12-19 15:03:...	2020-06-08	771.82	793.84	918.1	4.84
21411791	961	PARNAMIRIM	RN	4789004	false	2025-12-19 15:03:...	2020-05-22	588.82	817.98	1016.67	5.06
5881958	1018	PRESIDENTE DUTR	MA	47.71-7-01	false	2025-12-19 15:03:...	null	null	null	null	nul
58814862	1307	ANGICAL DO PIAU	PI	4771701	false	2025-12-19 15:03:...	2020-06-09	707.98	778.77	765.14	41.95
2149478	1391	TERESINA	PI	8122200	false	2025-12-19 15:03:...	2020-06-19	344.02	301.84	329.76	2.6
21437159	1581	FARIAS BRITO	CE	8122200	false	2025-12-19 15:03:...	2020-02-04	null	54.01	44.11	nul
58814851	1598	SAO RAIMUNDO NO	PI	4771701	false	2025-12-19 15:03:...	null	null	null	null	nul
58814274	1718	CASTELO DO PIAU	PI	4771701	false	2025-12-19 15:03:...	2020-06-25	338.09	375.12	346.66	15.32
5886297	1743	ALTAMIRA	PA	47.71-7-01	false	2025-12-19 15:03:...	2020-06-19	613.83	757.09	836.48	18.07
21411956	1889	MOGEIRO	PB	0151201	false	2025-12-19 15:03:...	2019-10-22	null	null	1317.18	nul
58811830	2039	ITABAIANA	PB	4771-7/01	false	2025-12-19 15:03:...	2020-06-23	497.72	686.28	692.51	11.76
2147032	2352	BREJO	MA	4712100	false	2025-12-19 15:03:...	null	null	null	null	nul



```
wd = '/delta'
```

In [2]:

```python
from pyspark.sql import SparkSession
import pandas as pd

# Cria uma sessão Spark
spark = SparkSession.builder \
    .appName("Ler Parquet do HDFS e converter para pandas") \
    .getOrCreate()

# Caminho para o diretório Parquet no HDFS
caminho_hdfs = "hdfs://localhost:8020/delta/data/gold-parquet"

# Lê todos os arquivos Parquet no diretório e cria um DataFrame do Spark
df_spark = spark.read.parquet(caminho_hdfs)

# Exibe o schema do DataFrame do Spark
df_spark.printSchema()

# Converte o DataFrame do Spark para um DataFrame do pandas
# Nota: Isso pode consumir muita memória, dependendo do tamanho do DataFrame
df_pandas = df_spark.toPandas()

# Exibe o DataFrame do pandas
print(df_pandas)
```

```
root
 |-- client_id: string (nullable = true)
 |-- key: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- cnae_id: string (nullable = true)
 |-- defaulting: string (nullable = true)
 |-- timestamp: timestamp (nullable = true)
 |-- ultima_compra: date (nullable = true)
 |-- pedidos_4_meses: double (nullable = true)
 |-- pedidos_8_meses: double (nullable = true)
 |-- pedidos_12_meses: double (nullable = true)
 |-- itens_4_meses: double (nullable = true)
 |-- itens_8_meses: double (nullable = true)
 |-- itens_12_meses: double (nullable = true)

       client_id    key              city state    cnae_id defaulting  \
0       21410134     95          SAO LUIS    MA    4789004      false
1        5886004    408             VIGIA    PA    4771701      false
2        5889897    412       JOAO PESSOA    PB    4771701      false
3       58822433    471       ESPERANTINA    PI    4771701      false
4        2142214    485   SANTA QUITERIA    CE    4789004      false
...          ...    ...               ...   ...        ...        ...
14995   58813415  13309  NOSSA SENHORA D    PI    4771701       true
14996   58821199  13476          PIRIPIRI    PI    4771701       true
14997   20536821  13582         BOA VISTA    RR       None       true
14998  588121289  13652       ULIANOPOLIS    PA       None       true
```

```
root
 |-- client_id: string (nullable = true)
 |-- key: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- cnae_id: string (nullable = true)
 |-- defaulting: string (nullable = true)
 |-- timestamp: timestamp (nullable = true)
 |-- ultima_compra: date (nullable = true)
 |-- pedidos_4_meses: double (nullable = true)
 |-- pedidos_8_meses: double (nullable = true)
 |-- pedidos_12_meses: double (nullable = true)
 |-- itens_4_meses: double (nullable = true)
 |-- itens_8_meses: double (nullable = true)
 |-- itens_12_meses: double (nullable = true)

       client_id    key            city state    cnae_id defaulting  \
0       21410134     95        SAO LUIS    MA    4789004      false
1        5886004    408           VIGIA    PA    4771701      false
2        5889897    412     JOAO PESSOA    PB    4771701      false
3       58822433    471     ESPERANTINA    PI    4771701      false
4        2142214    485   SANTA QUITERIA    CE    4789004      false
...          ...    ...             ...   ...        ...        ...
14995   58813415  13309  NOSSA SENHORA D    PI    4771701       true
14996   58821199  13476        PIRIPIRI    PI    4771701       true
14997   20536821  13582       BOA VISTA    RR       None       true
14998  588121289  13652     ULIANOPOLIS    PA       None       true
14999   58821975  14342  CAMPINA GRANDE    PB 47.71-7-01       None

                    timestamp ultima_compra  pedidos_4_meses  pedidos_8_meses  \
0     2025-12-19 15:03:31.789          None              NaN              NaN
1     2025-12-19 15:03:31.819          None              NaN              NaN
2     2025-12-19 15:03:31.819    2020-06-23           462.82           431.63
3     2025-12-19 15:03:31.820    2020-06-25           211.64           211.64
4     2025-12-19 15:03:31.820          None              NaN              NaN
...                       ...           ...              ...              ...
14995 2025-12-19 15:03:33.112    2020-06-16           811.79           854.88
14996 2025-12-19 15:03:33.113    2020-06-18          5452.44          5452.44
14997 2025-12-19 15:03:33.113          None              NaN              NaN
14998 2025-12-19 15:03:33.113          None              NaN              NaN
14999 2025-12-19 15:03:33.116    2020-06-17           297.31           297.31

       pedidos_12_meses  itens_4_meses  itens_8_meses  itens_12_meses
0                   NaN            NaN            NaN             NaN
1                   NaN            NaN            NaN             NaN
2                431.63          11.94          11.34           11.34
3                211.64           9.79           9.79            9.79
4                   NaN            NaN            NaN             NaN
...                 ...            ...            ...             ...
14995            817.10          24.32          22.45           20.43
14996           1953.50           9.29           9.29            6.95
14997               NaN            NaN            NaN             NaN
14998               NaN            NaN            NaN             NaN
```

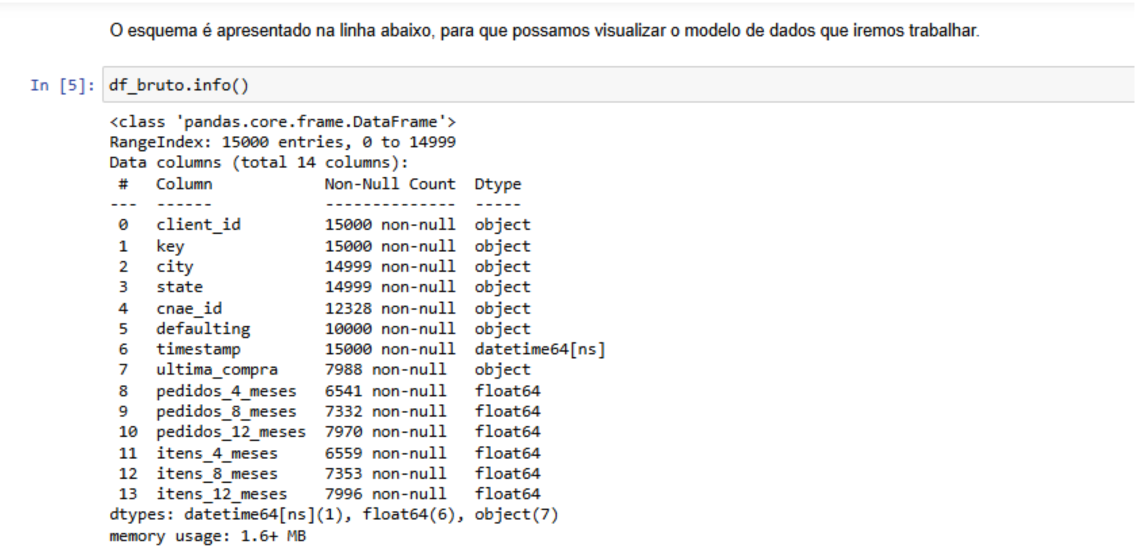
Leitura dos dados

O trecho de código abaixo cria uma variável *work_dir*, que irá apontar para o caminho no sistema de arquivos onde estão os dados de entrada e onde a saída será escrita. Como os dados de entrada estão no formato Parquet, o Pandas irá utilizar o motor de leitura Pyarrow para conseguir ler este formato de dados e aumentar a performance de leitura e transformações no DataFrame.

```
In [4]: df_bruto.head()
```

Out[4]:

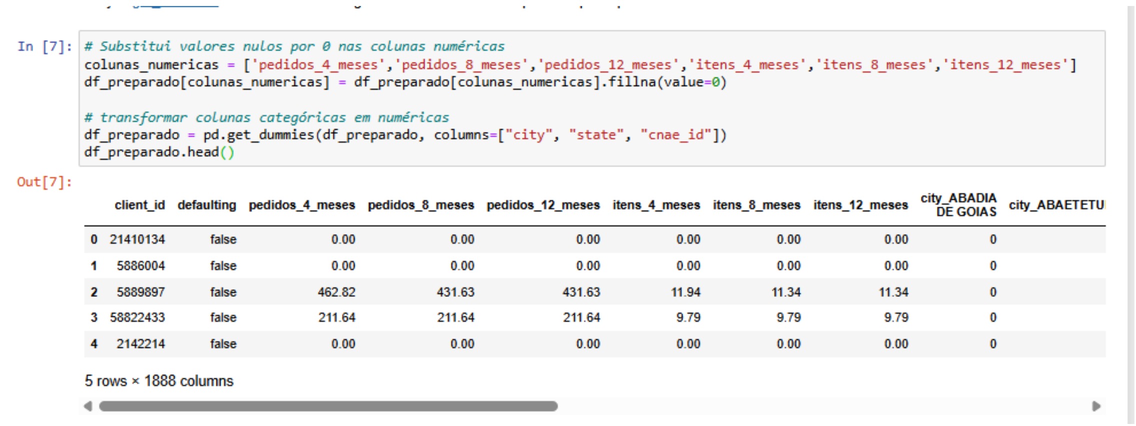
	client_id	key	city	state	cnae_id	defaulting	timestamp	ultima_compra	pedidos_4_meses	pedidos_8_meses	pedidos_12_meses	itens_4_mes
0	21410134	95	SAO LUIS	MA	4789004	false	2025-12-19 15:03:31.789	None	NaN	NaN	NaN	N
1	5886004	408	VIGIA	PA	4771701	false	2025-12-19 15:03:31.819	None	NaN	NaN	NaN	N
2	5889897	412	JOAO PESSOA	PB	4771701	false	2025-12-19 15:03:31.819	2020-06-23	462.82	431.63	431.63	11.
3	58822433	471	ESPERANTINA	PI	4771701	false	2025-12-19 15:03:31.820	2020-06-25	211.64	211.64	211.64	9.
4	2142214	485	SANTA QUITERIA	CE	4789004	false	2025-12-19 15:03:31.820	None	NaN	NaN	NaN	N



O esquema é apresentado na linha abaixo, para que possamos visualizar o modelo de dados que iremos trabalhar.

```
In [5]: df_bruto.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15000 entries, 0 to 14999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   client_id        15000 non-null  object
 1   key              15000 non-null  object
 2   city             14999 non-null  object
 3   state            14999 non-null  object
 4   cnae_id          12328 non-null  object
 5   defaulting       10000 non-null  object
 6   timestamp        15000 non-null  datetime64[ns]
 7   ultima_compra    7988 non-null   object
 8   pedidos_4_meses  6541 non-null   float64
 9   pedidos_8_meses  7332 non-null   float64
 10  pedidos_12_meses 7970 non-null   float64
 11  itens_4_meses    6559 non-null   float64
 12  itens_8_meses    7353 non-null   float64
 13  itens_12_meses   7996 non-null   float64
dtypes: datetime64[ns](1), float64(6), object(7)
memory usage: 1.6+ MB
```

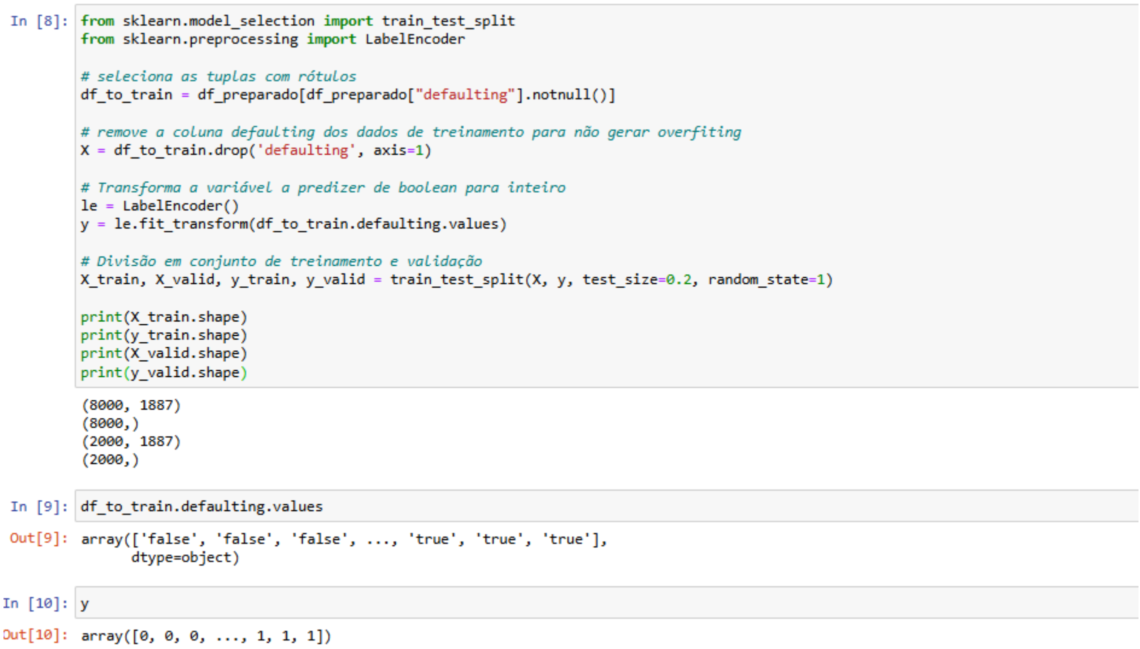


```
In [7]: # Substitui valores nulos por 0 nas colunas numéricas
        colunas_numericas = ['pedidos_4_meses','pedidos_8_meses','pedidos_12_meses','itens_4_meses','itens_8_meses','itens_12_meses']
        df_preparado[colunas_numericas] = df_preparado[colunas_numericas].fillna(value=0)

        # transformar colunas categóricas em numéricas
        df_preparado = pd.get_dummies(df_preparado, columns=["city", "state", "cnae_id"])
        df_preparado.head()
```

Out[7]:

	client_id	defaulting	pedidos_4_meses	pedidos_8_meses	pedidos_12_meses	itens_4_meses	itens_8_meses	itens_12_meses	city_ABADIA DE GOIAS	city_ABAETETU
0	21410134	false	0.00	0.00	0.00	0.00	0.00	0.00	0	
1	5886004	false	0.00	0.00	0.00	0.00	0.00	0.00	0	
2	5889897	false	462.82	431.63	431.63	11.94	11.34	11.34	0	
3	58822433	false	211.64	211.64	211.64	9.79	9.79	9.79	0	
4	2142214	false	0.00	0.00	0.00	0.00	0.00	0.00	0	

5 rows × 1888 columns



```
In [8]: from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import LabelEncoder

        # seleciona as tuplas com rótulos
        df_to_train = df_preparado[df_preparado["defaulting"].notnull()]

        # remove a coluna defaulting dos dados de treinamento para não gerar overfiting
        X = df_to_train.drop('defaulting', axis=1)

        # Transforma a variável a predizer de boolean para inteiro
        le = LabelEncoder()
        y = le.fit_transform(df_to_train.defaulting.values)

        # Divisão em conjunto de treinamento e validação
        X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.2, random_state=1)

        print(X_train.shape)
        print(y_train.shape)
        print(X_valid.shape)
        print(y_valid.shape)
```

```
(8000, 1887)
(8000,)
(2000, 1887)
(2000,)
```

```
In [9]: df_to_train.defaulting.values
Out[9]: array(['false', 'false', 'false', ..., 'true', 'true', 'true'],
              dtype=object)
```

```
In [10]: y
Out[10]: array([0, 0, 0, ..., 1, 1, 1])
```

Treinamento e Avaliação do modelo

Nesta etapa iremos treinar o nosso classificador, neste caso uma árvore de decisão. Os dados de treinamento estão armazenados em *X_train* (features) e *y_train* (rótulo). A predição é realizada com os dados de treinamento em *X_valid*.

In [11]:

```
from sklearn.tree import DecisionTreeClassifier

# Cria um classificador
clf = DecisionTreeClassifier()

# Treina a Árvore de Decisão
clf = clf.fit(X_train,y_train)

# Prediz a resposta para o dataset de validação
y_pred = clf.predict(X_valid)
```

Para **avaliar** a acurácia do modelo, o resultado da predição *y_pred* é comparado com o resultado esperado *y_valid* para gerar as métricas ROC, Acurácia e F1.

In [12]:

```
import sklearn.metrics as metrics
print("ROC AUC:",metrics.roc_auc_score(y_valid, y_pred))
print("Acurácia:",metrics.accuracy_score(y_valid, y_pred))
print("F1 score:",metrics.f1_score(y_valid, y_pred))
```

```
ROC AUC: 0.7728781412991939
Acurácia: 0.816
F1 score: 0.6827586206896552
```

Predição sobre os dados de testes

Nesta última etapa, o modelo busca predizer se o cliente está ou não inadimplente sobre os dados de teste (coluna defaulting igual a nulo). Os dados de teste ficarão armazenados no DataFrame *df_test*.

In [13]:

```
df_test = df_preparado[df_preparado["defaulting"].isnull()]
df_test.shape
```

Out[13]:

```
(5000, 1888)
```

Os dados de teste são gerados e armazenados em *X_test*, excluindo a coluna *defaulting* que desejamos predizer. O modelo faz a predição e tem como saída os valores da predição em *y_test*.

In [14]:

```
X_test = df_test.drop('defaulting', axis=1)
y_test = clf.predict(X_test)
y_test
```

Out[14]:

```
array([0, 0, 0, ..., 0, 0, 0])
```

A saída do modelo é salvo em um arquivo csv, contendo as colunas "client_id" e "inadimplente". Estas colunas serão utilizadas para avaliar a acurácia do modelo. Por isso, o resultado da predição em *y_test* é adicionada em uma nova coluna (inadimplente) do DataFrame df_test.

In [15]:

```
output = df_test.assign(inadimplente=y_test)
output = output.loc[:, ['client_id','inadimplente']]
output.head()
```

Out[15]:

	client_id	inadimplente
35	58813073	0
36	58819272	0
41	2143865	0
42	58813592	0
43	2145388	1

O Dataframe *output* é escrito no formato CSV para gerar a saída do algoritmo de aprendizado de máquina construído neste notebook.

In [17]:

```
output.to_csv("/home/bigdata/ouput_sklearn.csv", index=False)
```

Considerações Finais

Agora é com **você**! Ainda existe muito espaço para melhoria na acurácia do modelo que desenvolvemos até agora. Utilize o material complementar abaixo para modificar este notebook e construir um algoritmo melhor.

- Curso de Aprendizado de Máquina de Stanford com Andrew Ng
- Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow
- Introduction to Machine Learning with Python
- Data Science do Zero
- Customer Churn Classification Using Predictive Machine Learning Models