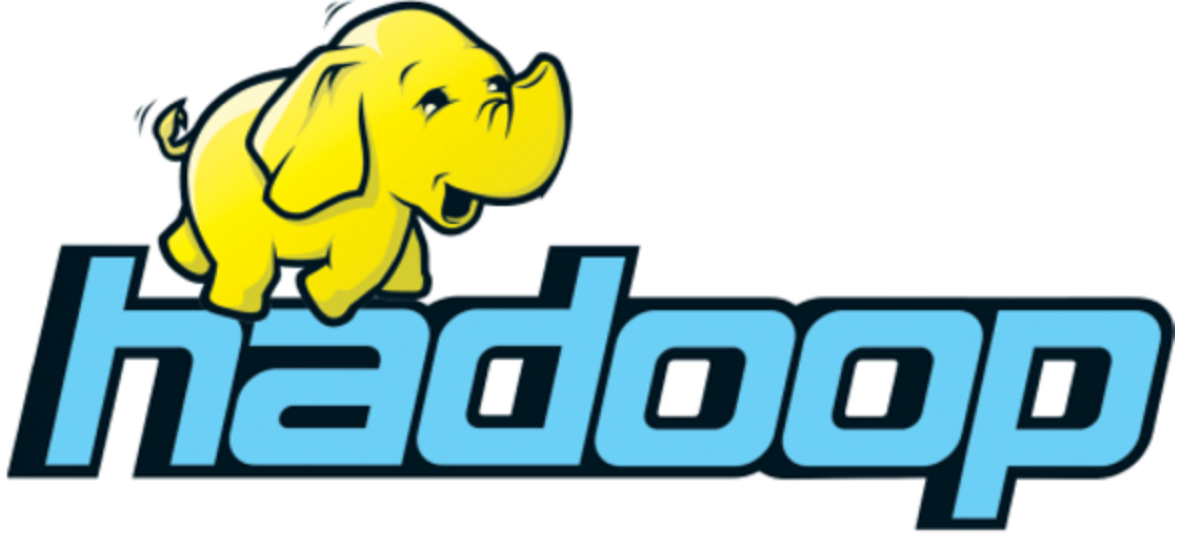

hadoop

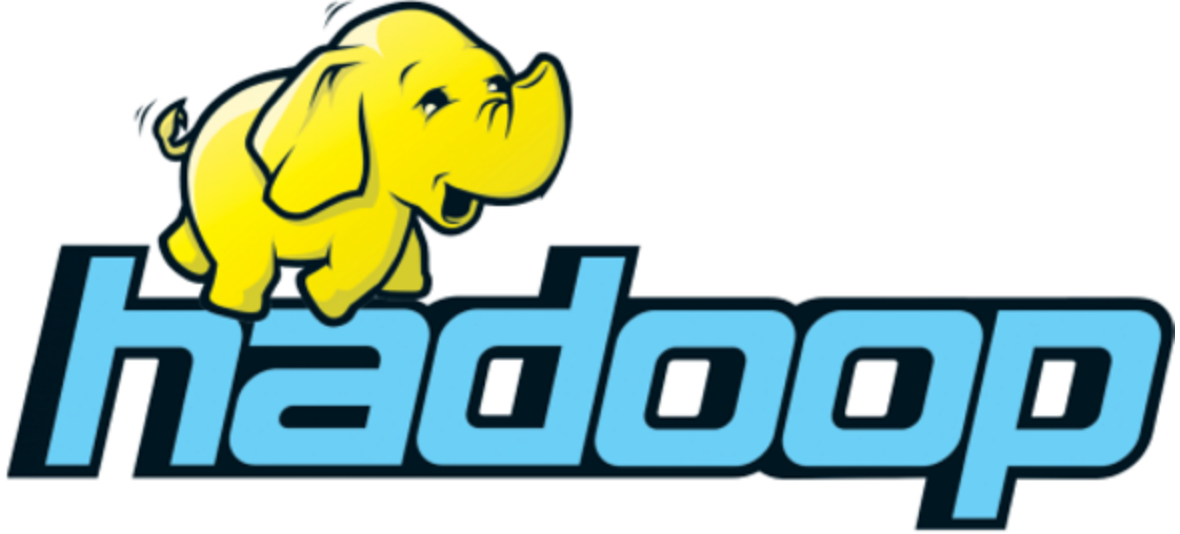


Partições

Compressão

Bucketing

Windowing Functions



Partições

Bucketing

Otimização de Performance

Compressão

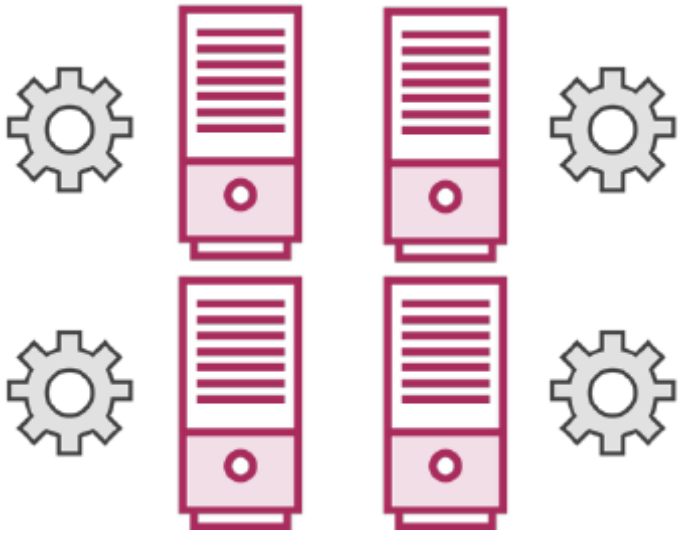
Windowing Functions

Boas práticas

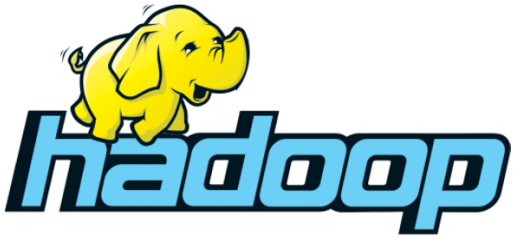
DataWarehouse

Dados são armazenados em múltiplas máquinas do cluster.

Framework Distribuído



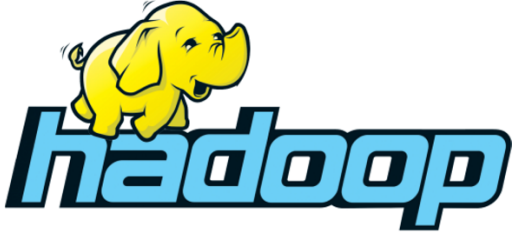
Fundação Apache – Open Source



Sistema de arquivos

Paradigma de computação paralela

Gerenciador



HIVE Query Language

- Namenodes
- Datanodes
  - Replicação
  - Criação

```
user
└── hive
    └── applestore
        ├── data
        │   └── file-01
        └── example_table
            ├── file-01
            └── file-02
```

Access via:

/user/hive/applestore/data/file-01

```
user
└── hive
    └── applestore
        ├── data
        │   └── file-01
        └── example_table
            ├── file-01
            └── file-02
```


```
user
└── hive
    └── applestore
        ├── data
        │   └── file-01
        └── example_table
            ├── file-01
            └── file-02
```

Partições

São usadas para acelerar cláusula WHERE

Partições são PASTAS no HDFS

Partições

pk_cd_cnpj	str_name	cat_segmento	vlr_transacao	qtd_transac	cat_estado
1	SANTADER	K	1000	599	SP
2	UBER	K	25000	58	RJ
3	ITAU	2	400000	66	RJ
4	IFOOD	D	100000	99	SP
5	RAPPI	2	500000	199	PR
6	NUBANK	K	25000	88	ES
7	GOOGLE	D	400000	45	ES
8	FACEBOOK	D	500000	25	PR

SELECT * FROM table WHERE CAT_ESTADO=“SP”

Partições

SP

PR

RJ

ES

Partições

Partições

/user/hive/applestore/data/

Partições

/user/hive/applestore/data/

/cat_estado=SP

/cat_estado=RJ

/cat_estado=PR

/cat_estado=ES

Partições

/user/hive/applestore/data/

/cat_estado=SP

/file-01

Partições

/user/hive/applestore/data/cat_estado=SP/file-01

pk_cd_cnpj	str_name	cat_segmento	vlr_transacao	qtd_transacao	cat_estado
1	SANTADER	K	1000	599	SP
4	IFOOD	D	100000	99	SP

cat_estado=SP

pk_cd_cnpj	str_name	cat_segmento	vlr_transacao	qtd_transacao	cat_estado
2	UBER	K	25000	58	RJ
3	ITAU	2	400000	66	RJ

cat_estado=RJ

pk_cd_cnpj	str_name	cat_segmento	vlr_transacao	qtd_transacao	cat_estado
5	RAPPI	2	500000	199	PR
8	FACEBOOK	D	500000	25	PR

cat_estado=PR

pk_cd_cnpj	str_name	cat_segmento	vlr_transacao	qtd_transacao	cat_estado
6	NUBANK	K	25000	88	ES
7	GOOGLE	D	400000	45	ES

cat_estado=ES

Partições

- Devem ser aplicadas à coluna que você queira filtrar mais frequentemente

- É possível fazer mais de uma partição

- Balanceamento de carga

- Milhares de partições podem trazer grande overhead

Compressão

Compressão

Row Storage

1	SANTADER	K	1000	599	SP
2	UBER	K	25000	58	RJ
3	ITAU	2	400000	66	RJ
4	IFOOD	D	100000	99	SP

Column Storage

pk_cd_cnpj	cat_segmento	vlr_transacao	cat_estado
1	K	1000	SP
2	K	25000	RJ
3	2	400000	RJ
4	D	100000	SP
5	2	500000	PR
6	K	25000	ES
7	D	400000	ES
8	D	500000	PR

Compressão

Row Storage

1	SANTADER	K	1000	599	SP
2	UBER	K	25000	58	RJ
3	ITAU	2	400000	66	RJ
4	IFOOD	D	100000	99	SP

Column Storage

pk_cd_cnpj
1
2
3
4
5
6
7
8

cat_segmento
K
K
2
D
2
K
D
D

vlr_transacao
1000
25000
400000
100000
500000
25000
400000
500000

cat_estado
SP
RJ
RJ
SP
PR
ES
ES
PR

Mais fácil de comprimir!

vlr_transacao
0
...
0
1000
NaN
...
NaN
NaN

500x0-1x1000-400xNaN

Compressão

Row Storage

1	SANTADER	K	1000	599	SP
2	UBER	K	25000	58	RJ
3	ITAU	2	400000	66	RJ
4	IFOOD	D	100000	99	SP

Column Storage

pk_cd_cnpj
1
2
3
4
5
6
7
8

cat_segmento
K
K
2
D
2
K
D
D

vlr_transacao
1000
25000
400000
100000
500000
25000
400000
500000

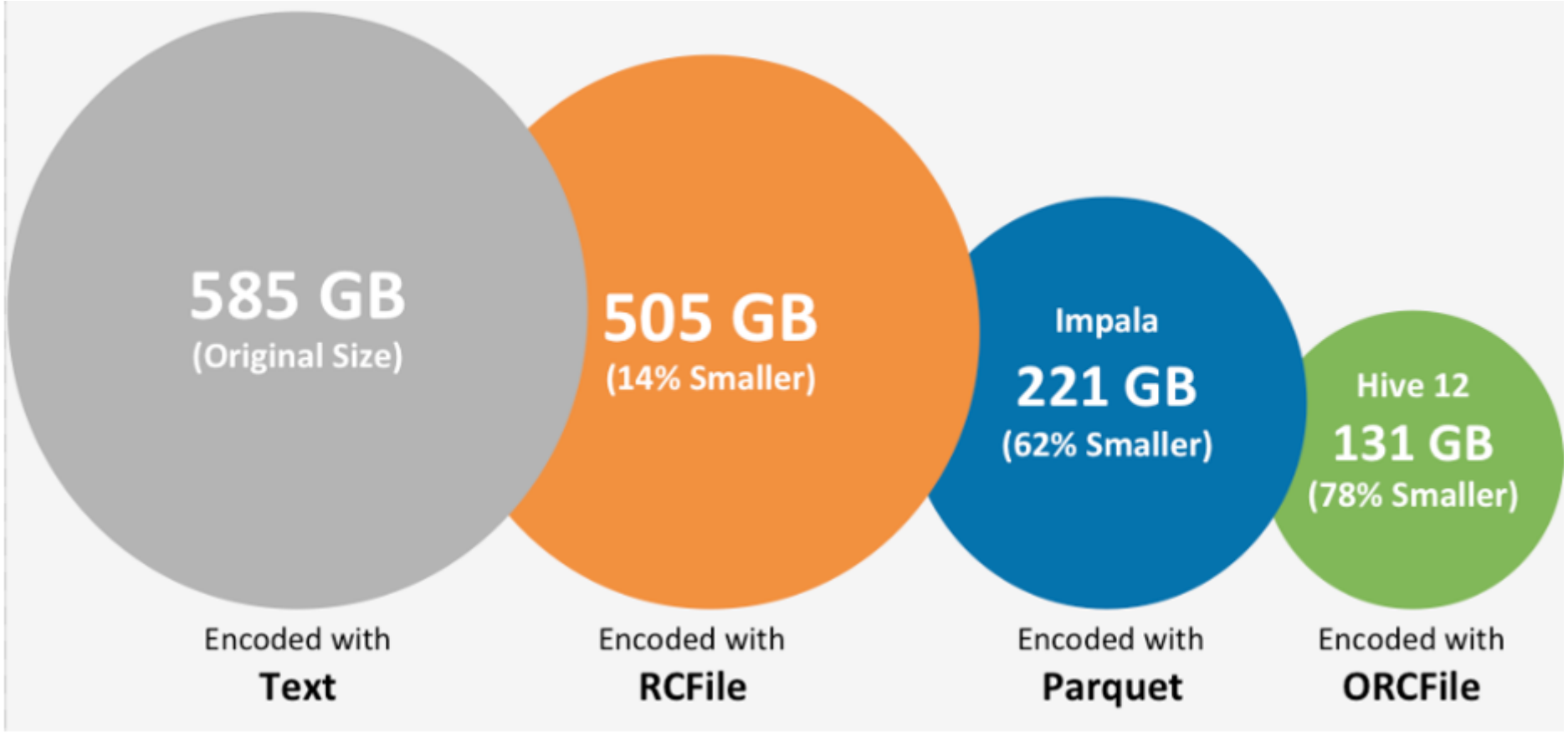
cat_estado
SP
RJ
RJ
SP
PR
ES
ES
PR

Mais fácil de comprimir!

fl_target
0
1
1
0
0
0
1
0

Compressão

Comparação



Bucketing

Bucketing

São usadas para acelerar cláusula JOINS

Partições são ARQUIVOS no HDFS

Bucketing é o equivalente lógico de Hash

Bucketing

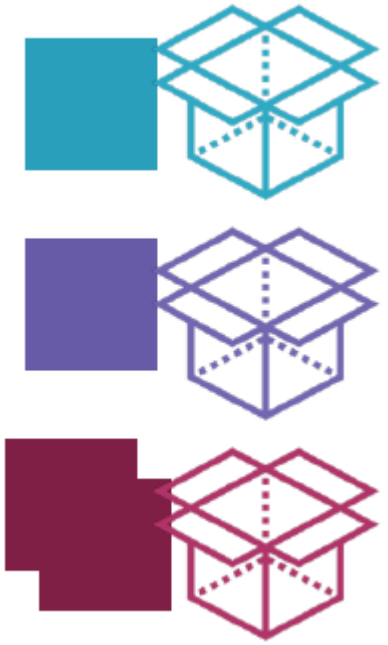
Bucketing

Bucketing

Bucketing

Bucketing

Bucketing



Bucketing

pk_cd_cnpj	str_name	cat_segmento	vlr_transacao	qtd_transac	cat_estado
1	SANTADER	K	1000	599	SP
2	UBER	K	25000	58	RJ
3	ITAU	2	400000	66	RJ
4	IFOOD	D	100000	99	SP
5	RAPPI	2	500000	199	PR
6	NUBANK	K	25000	88	ES
7	GOOGLE	D	400000	45	ES
8	FACEBOOK	D	500000	25	PR

3 Buckets – pk_cd_cnpj

Para inteiros, a função de hash é simples (módulo %)

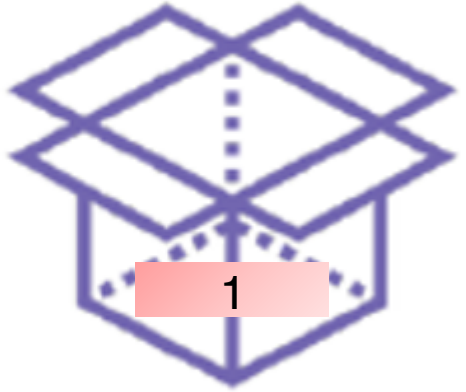
Bucketing

pk_cd_cnpj	str_name	cat_segmento	vlr_transacao	qtd_transac	cat_estado
1	SANTADER	K	1000	599	SP
2	UBER	K	25000	58	RJ
3	ITAU	2	400000	66	RJ
4	IFOOD	D	100000	99	SP
5	RAPPI	2	500000	199	PR
6	NUBANK	K	25000	88	ES
7	GOOGLE	D	400000	45	ES
8	FACEBOOK	D	500000	25	PR

1 % 3 = 1

0

1



2

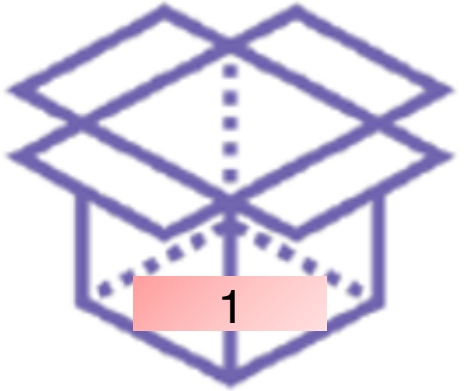
Bucketing

pk_cd_cnpj	str_name	cat_segmento	vlr_transacao	qtd_transac	cat_estado
1	SANTADER	K	1000	599	SP
2	UBER	K	25000	58	RJ
3	ITAU	2	400000	66	RJ
4	IFOOD	D	100000	99	SP
5	RAPPI	2	500000	199	PR
6	NUBANK	K	25000	88	ES
7	GOOGLE	D	400000	45	ES
8	FACEBOOK	D	500000	25	PR

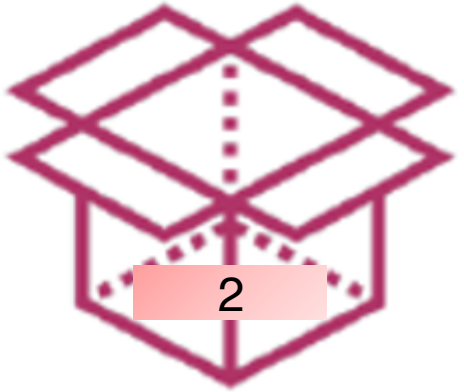
2 % 3 = 2

0

1



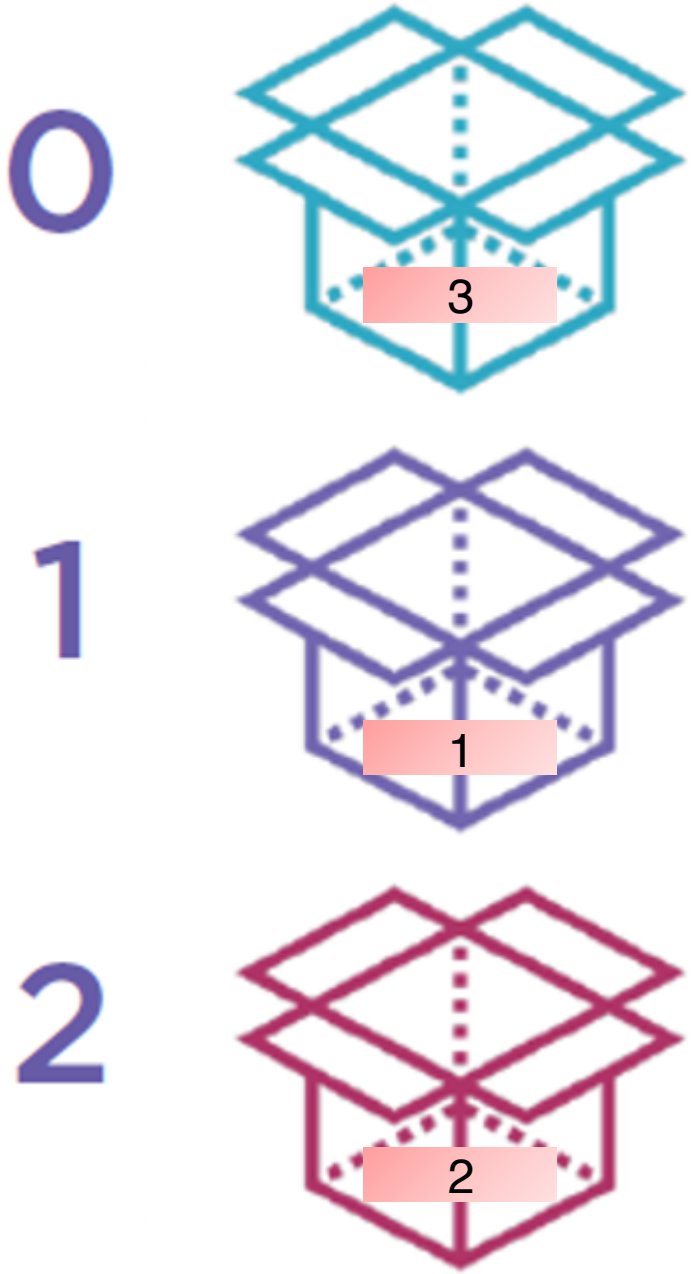
2



Bucketing

pk_cd_cnpj	str_name	cat_segmento	vlr_transacao	qtd_transac	cat_estado
1	SANTADER	K	1000	599	SP
2	UBER	K	25000	58	RJ
3	ITAU	2	400000	66	RJ
4	IFOOD	D	100000	99	SP
5	RAPPI	2	500000	199	PR
6	NUBANK	K	25000	88	ES
7	GOOGLE	D	400000	45	ES
8	FACEBOOK	D	500000	25	PR

3 % 3 = 0



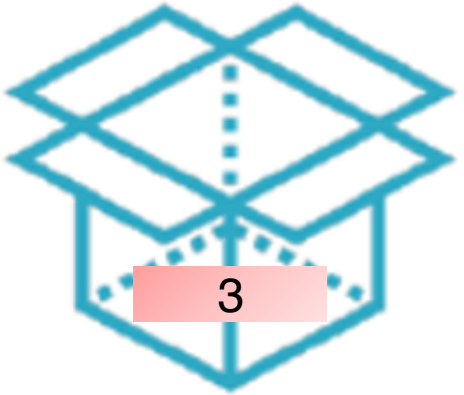
Bucketing

pk_cd_cnpj	str_name	cat_segmento	vlr_transacao	qtd_transac	cat_estado
1	SANTADER	K	1000	599	SP
2	UBER	K	25000	58	RJ
3	ITAU	2	400000	66	RJ
4	IFOOD	D	100000	99	SP
5	RAPPI	2	500000	199	PR
6	NUBANK	K	25000	88	ES
7	GOOGLE	D	400000	45	ES
8	FACEBOOK	D	500000	25	PR

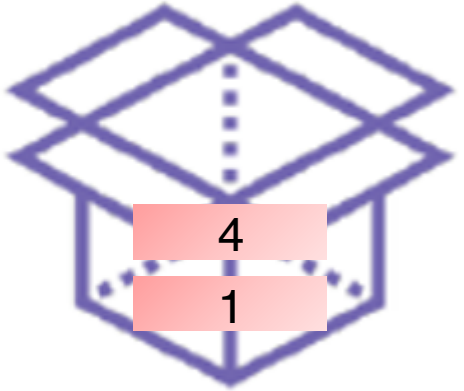
4 % 3 =

1

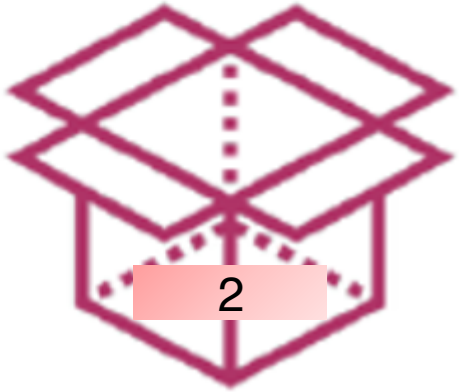
0



1



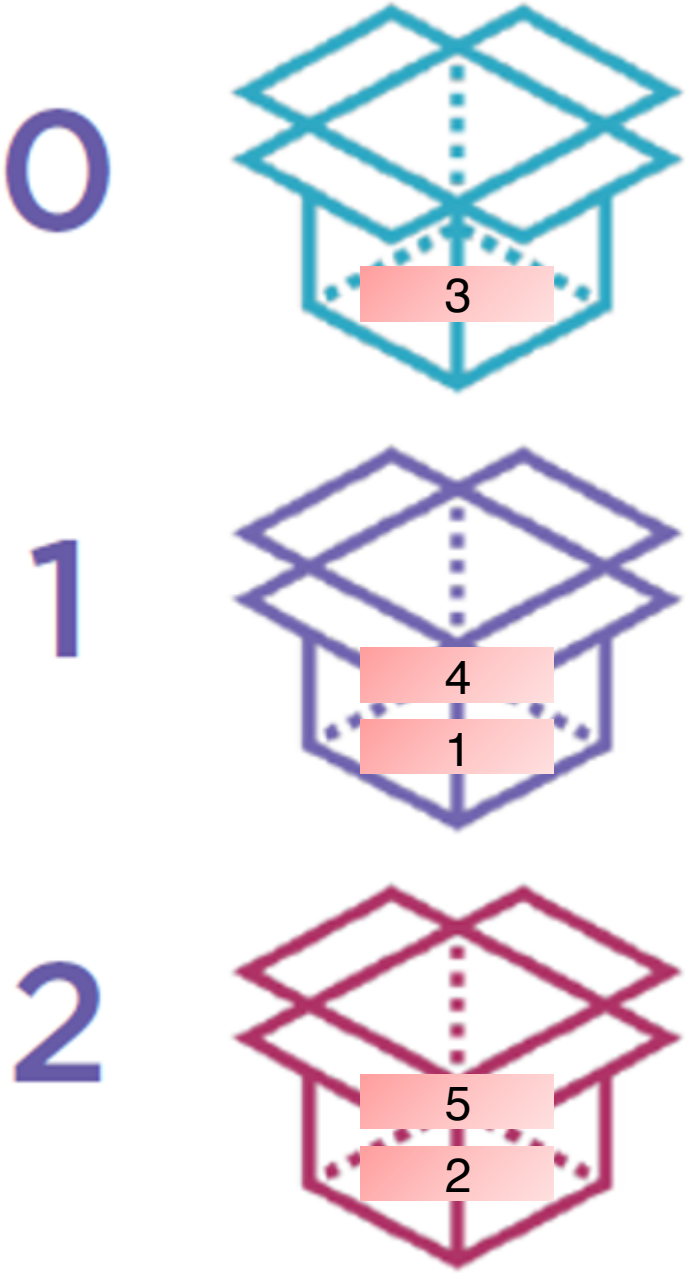
2



Bucketing

pk_cd_cnpj	str_name	cat_segmento	vlr_transacao	qtd_transac	cat_estado
1	SANTADER	K	1000	599	SP
2	UBER	K	25000	58	RJ
3	ITAU	2	400000	66	RJ
4	IFOOD	D	100000	99	SP
5	RAPPI	2	500000	199	PR
6	NUBANK	K	25000	88	ES
7	GOOGLE	D	400000	45	ES
8	FACEBOOK	D	500000	25	PR

5 % 3 =
2

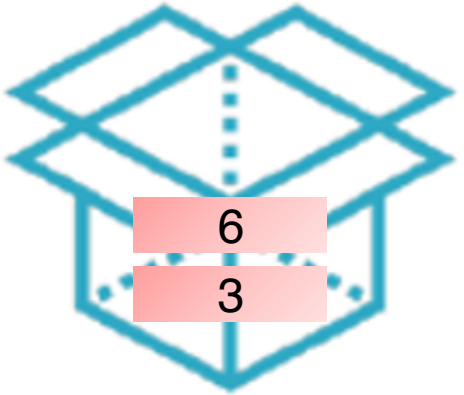


Bucketing

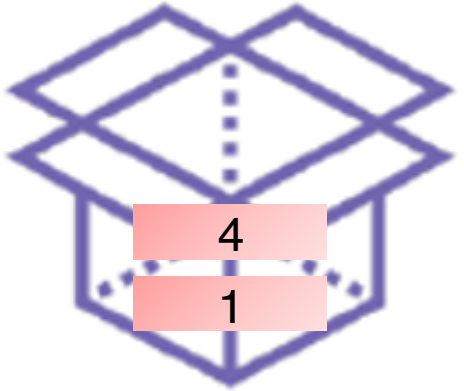
pk_cd_cnpj	str_name	cat_segmento	vlr_transacao	qtd_transac	cat_estado
1	SANTADER	K	1000	599	SP
2	UBER	K	25000	58	RJ
3	ITAU	2	400000	66	RJ
4	IFOOD	D	100000	99	SP
5	RAPPI	2	500000	199	PR
6	NUBANK	K	25000	88	ES
7	GOOGLE	D	400000	45	ES
8	FACEBOOK	D	500000	25	PR

6 % 3 =
0

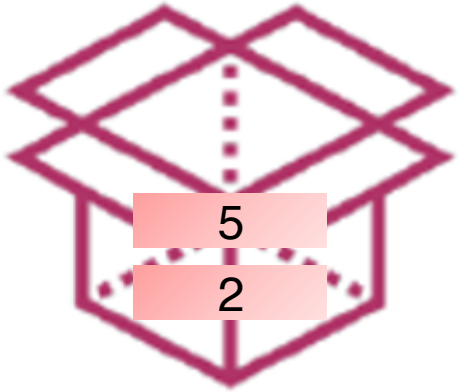
0



1



2



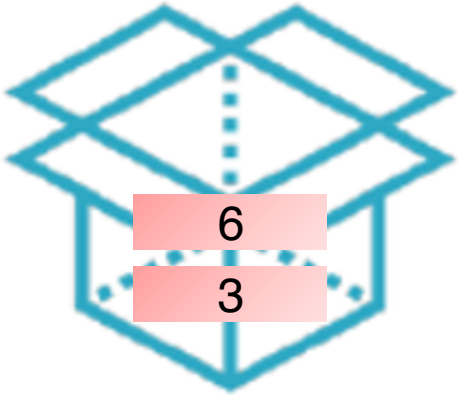
Bucketing

pk_cd_cnpj	str_name	cat_segmento	vlr_transacao	qtd_transac	cat_estado
1	SANTADER	K	1000	599	SP
2	UBER	K	25000	58	RJ
3	ITAU	2	400000	66	RJ
4	IFOOD	D	100000	99	SP
5	RAPPI	2	500000	199	PR
6	NUBANK	K	25000	88	ES
7	GOOGLE	D	400000	45	ES
8	FACEBOOK	D	500000	25	PR

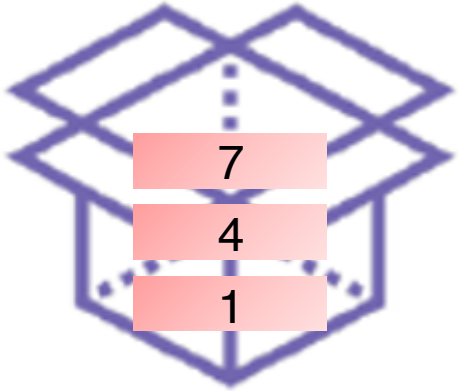
7 % 3 =

1

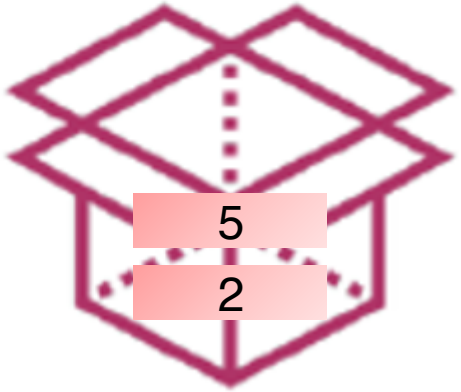
0



1



2



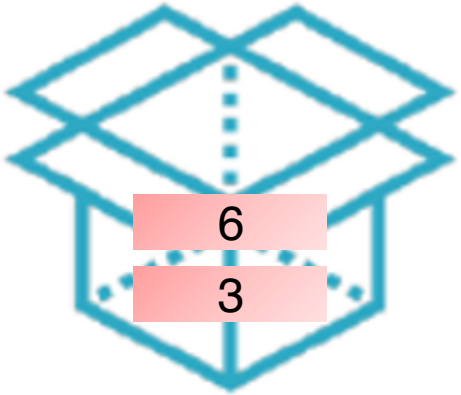
Bucketing

pk_cd_cnpj	str_name	cat_segmento	vlr_transacao	qtd_transac	cat_estado
1	SANTADER	K	1000	599	SP
2	UBER	K	25000	58	RJ
3	ITAU	2	400000	66	RJ
4	IFOOD	D	100000	99	SP
5	RAPPI	2	500000	199	PR
6	NUBANK	K	25000	88	ES
7	GOOGLE	D	400000	45	ES
8	FACEBOOK	D	500000	25	PR

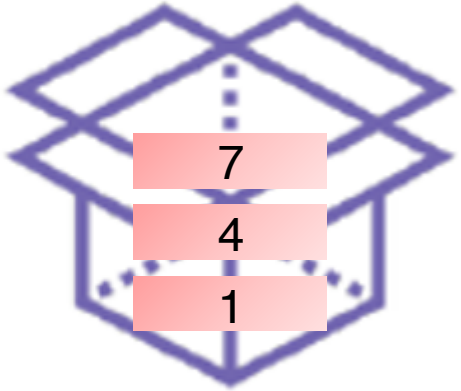
8 % 3 =

2

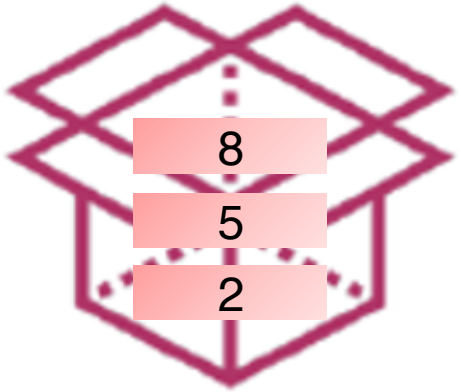
0



1



2



Bucketing

pk_cd_cnpj	str_name	cat_segmento	vlr_transacao	qtd_transac	cat_estado
1	SANTADER	K	1000	599	SP
2	UBER	K	25000	58	RJ
3	ITAU	2	400000	66	RJ
4	IFOOD	D	100000	99	SP
5	RAPPI	2	500000	199	PR
6	NUBANK	K	25000	88	ES
7	GOOGLE	D	400000	45	ES
8	FACEBOOK	D	500000	25	PR

pk_cd_cnpj	str_name	fl_ever30
8	FACEBOOK	500000
7	GOOGLE	400000
6	NUBANK	25000
5	RAPPI	500000
4	IFOOD	100000
3	ITAU	400000
2	UBER	25000
1	SANTADER	1000

Precisa percorrer a tabela inteira

Bucketing

pk_cd_cnpj	str_name	cat_segmento	vlr_transacao	qtd_transac	cat_estado
1	SANTADER	K	1000	599	SP
2	UBER	K	25000	58	RJ
3	ITAU	2	400000	66	RJ
4	IFOOD	D	100000	99	SP
5	RAPPI	2	500000	199	PR
6	NUBANK	K	25000	88	ES
7	GOOGLE	D	400000	45	ES
8	FACEBOOK	D	500000	25	PR

pk_cd_cnpj	str_name	fl_ever30
6	NUBANK	25000
3	ITAU	400000

0

pk_cd_cnpj	str_name	fl_ever30
7	GOOGLE	400000
4	IFOOD	100000
1	SANTADER	1000

1

pk_cd_cnpj	str_name	fl_ever30
8	FACEBOOK	500000
5	RAPPI	500000
2	UBER	25000

2

Precisa percorrer 1 bucket.

Window Functions

```
FUNCTION_NAME ( field )
 OVER (window_clause) AS alias
```

Window Functions

FUNCTION_NAME (field)
OVER (**window_clause**) AS alias

firstname	department	startdate
Andrew	1	1/23/1999
Jacob	1	7/11/1990
Daniel	2	6/24/2004
Anna	1	10/7/2001
Pierre	1	2/22/2009
Ruth	2	6/6/1998
Anthony	1	11/29/1995
Isabella	2	9/28/1997
Jose	2	3/17/2013

Window Functions

```
SELECT firstname,
       department,
       startdate,
       RANK() OVER (PARTITION BY
department ORDER BY startdate ) AS rank
 FROM Employees;
```

firstname	department	startdate
Andrew	1	1/23/1999
Jacob	1	7/11/1990
Daniel	2	6/24/2004
Anna	1	10/7/2001
Pierre	1	2/22/2009
Ruth	2	6/6/1998
Anthony	1	11/29/1995
Isabella	2	9/28/1997
Jose	2	3/17/2013

Window Functions

PARTITION BY department

ORDER BY startdate

RANK()

firstname	department	startdate
Andrew	1	1/23/1999
Jacob	1	7/11/1990
Daniel	2	6/24/2004
Anna	1	10/7/2001
Pierre	1	2/22/2009
Ruth	2	6/6/1998
Anthony	1	11/29/1995
Isabella	2	9/28/1997
Jose	2	3/17/2013

firstname	department	startdate
Andrew	1	1/23/1999
Jacob	1	7/11/1990
Anna	1	10/7/2001
Pierre	1	2/22/2009
Anthony	1	11/29/1995

firstname	department	startdate
Ruth	2	6/6/1998
Daniel	2	6/24/2004
Jose	2	3/17/2013
Isabella	2	9/28/1997

firstname	department	startdate
Jacob	1	7/11/1990
Anthony	1	11/29/1995
Andrew	1	1/23/1999
Anna	1	10/7/2001
Pierre	1	2/22/2009

firstname	department	startdate
Isabella	2	9/28/1997
Daniel	2	6/24/2004
Jose	2	3/17/2013
Ruth	2	6/6/2013

firstname	department	startdate	rank
Jacob	1	7/11/1990	1
Anthony	1	11/29/1995	2
Andrew	1	1/23/1999	3
Anna	1	10/7/2001	4
Pierre	1	2/22/2009	5

firstname	department	startdate	rank
Isabella	2	9/28/1997	1
Daniel	2	6/24/2004	2
Jose	2	3/17/2013	3
Ruth	2	6/6/2013	4

Window Functions

```
SELECT *,
        AVG(vlr_transacao) OVER (ORDER
BY date ROWS BETWEEN 2 PRECEDING
AND CURRENT ROW) AS moving_avg
FROM t_employee;
```

```
'80389123000101',  4500, 201901
'80389123000103',  1500, 201901
'80389123000104',   500, 201901
'80389123000102',  4300, 201901
'80389123000101',  4500, 201902
'80389123000103',  1500, 201902
'80389123000104',   500, 201902
'80389123000102',  4300, 201902
'80389123000101',  4500, 201903
'80389123000103',  1500, 201903
'80389123000104',   500, 201903
'80389123000102',  4300, 201903
'80389123000101',  4500, 201904
```

Window Functions

```
SELECT *,
        AVG(vlr_transacao) OVER (ORDER
BY date ROWS BETWEEN 2 PRECEDING
AND CURRENT ROW) AS moving_avg
FROM t_employee;
```

```
'80389123000101',  4500, 201901),
'80389123000103',  1500, 201901),
'80389123000104',   500, 201901),
'80389123000102',  4300, 201901),
'80389123000101',  4500, 201902),
'80389123000103',  1500, 201902),
'80389123000104',   500, 201902),
'80389123000102',  4300, 201902),
'80389123000101',  4500, 201903),
'80389123000103',  1500, 201903),
'80389123000104',   500, 201903),
'80389123000102',  4300, 201903),
'80389123000101',  4500, 201904),
```

Window Functions

```
SELECT *,
        AVG(vlr_transacao) OVER (ORDER
BY date ROWS BETWEEN 2 PRECEDING
AND CURRENT ROW) AS moving_avg
FROM t_employee;
```

```
'80389123000101',  4500, 201901),
'80389123000103',  1500, 201901),
'80389123000104',   500, 201901),
'80389123000102',  4300, 201901),
'80389123000101',  4500, 201902),
'80389123000103',  1500, 201902),
'80389123000104',   500, 201902),
'80389123000102',  4300, 201902),
'80389123000101',  4500, 201903),
'80389123000103',  1500, 201903),
'80389123000104',   500, 201903),
'80389123000102',  4300, 201903),
'80389123000101',  4500, 201904),
```

Window Functions

```
SELECT *,
        AVG(vlr_transacao) OVER (ORDER
BY date ROWS BETWEEN 2 PRECEDING
AND CURRENT ROW) AS moving_avg
FROM t_employee;
```

```
'80389123000101',  4500, 201901),
'80389123000103',  1500, 201901),
'80389123000104',   500, 201901),
'80389123000102',  4300, 201901),
'80389123000101',  4500, 201902),
'80389123000103',  1500, 201902),
'80389123000104',   500, 201902),
'80389123000102',  4300, 201902),
'80389123000101',  4500, 201903),
'80389123000103',  1500, 201903),
'80389123000104',   500, 201903),
'80389123000102',  4300, 201903),
'80389123000101',  4500, 201904),
```

Window Functions

```
SELECT *,
        AVG(vlr_transacao) OVER (ORDER
BY date ROWS BETWEEN 2 PRECEDING
AND CURRENT ROW) AS moving_avg
FROM t_employee;
```

```
'80389123000101',  4500, 201901),
'80389123000103',  1500, 201901),
'80389123000104',   500, 201901),
'80389123000102',  4300, 201901),
'80389123000101',  4500, 201902),
'80389123000103',  1500, 201902),
'80389123000104',   500, 201902),
'80389123000102',  4300, 201902),
'80389123000101',  4500, 201903),
'80389123000103',  1500, 201903),
'80389123000104',   500, 201903),
'80389123000102',  4300, 201903),
'80389123000101',  4500, 201904),
```

Window Functions

```
SELECT *,
        AVG(vlr_transacao) OVER (ORDER
BY date ROWS BETWEEN 2 PRECEDING
AND CURRENT ROW) AS moving_avg
FROM t_employee;
```

```
'80389123000101',  4500, 201901),
'80389123000103',  1500, 201901),
'80389123000104',   500, 201901),
'80389123000102',  4300, 201901),
'80389123000101',  4500, 201902),
'80389123000103',  1500, 201902),
'80389123000104',   500, 201902),
'80389123000102',  4300, 201902),
'80389123000101',  4500, 201903),
'80389123000103',  1500, 201903),
'80389123000104',   500, 201903),
'80389123000102',  4300, 201903),
'80389123000101',  4500, 201904),
```

Window Functions

```
SELECT *,
        AVG(vlr_transacao) OVER (ORDER
BY date ROWS BETWEEN 2 PRECEDING
AND CURRENT ROW) AS moving_avg
FROM t_employee;
```

```
'80389123000101',  4500, 201901),
'80389123000103',  1500, 201901),
'80389123000104',   500, 201901),
'80389123000102',  4300, 201901),
'80389123000101',  4500, 201902),
'80389123000103',  1500, 201902),
'80389123000104',   500, 201902),
'80389123000102',  4300, 201902),
'80389123000101',  4500, 201903),
'80389123000103',  1500, 201903),
'80389123000104',   500, 201903),
'80389123000102',  4300, 201903),
'80389123000101',  4500, 201904),
```

Window Functions

```
SELECT *,
        AVG(vlr_transacao) OVER (ORDER
BY date ROWS BETWEEN 2 PRECEDING
AND CURRENT ROW) AS moving_avg
FROM t_employee;
```

```
'80389123000101',  4500, 201901),
'80389123000103',  1500, 201901),
'80389123000104',   500, 201901),
'80389123000102',  4300, 201901),
'80389123000101',  4500, 201902),
'80389123000103',  1500, 201902),
'80389123000104',   500, 201902),
'80389123000102',  4300, 201902),
'80389123000101',  4500, 201903),
'80389123000103',  1500, 201903),
'80389123000104',   500, 201903),
'80389123000102',  4300, 201903),
'80389123000101',  4500, 201904),
```

Window Functions

```
SELECT *,
        AVG(vlr_transacao) OVER (ORDER
BY date ROWS BETWEEN 2 PRECEDING
AND CURRENT ROW) AS moving_avg
FROM t_employee;
```

```
'80389123000101',  4500, 201901),
'80389123000103',  1500, 201901),
'80389123000104',   500, 201901),
'80389123000102',  4300, 201901),
'80389123000101',  4500, 201902),
'80389123000103',  1500, 201902),
'80389123000104',   500, 201902),
'80389123000102',  4300, 201902),
'80389123000101',  4500, 201903),
'80389123000103',  1500, 201903),
'80389123000104',   500, 201903),
'80389123000102',  4300, 201903),
'80389123000101',  4500, 201904),
```

Window Functions

```
SELECT *,
        AVG(vlr_transacao) OVER
(PARTITION BY pk_cd_cnpj ORDER BY date
ROWS BETWEEN 2 PRECEDING AND
CURRENT ROW) AS moving_avg
FROM t_employee;
```

```
('80389123000101',  4300, 201901)
('80389123000101',  4400, 201902)
('80389123000101',  4500, 201903)
('80389123000101',  4900, 201904)
('80389123000101',  4400, 201905)


('80389123000103',  1600, 201901)
('80389123000103',  1200, 201902)
('80389123000103',  1100, 201903)
('80389123000103',  1300, 201904)
('80389123000103',  1300, 201905)
```

Window Functions

```
SELECT *,
        AVG(vlr_transacao) OVER
(PARTITION BY pk_cd_cnpj ORDER BY date
ROWS BETWEEN 2 PRECEDING AND
CURRENT ROW) AS moving_avg
FROM t_employee;
```

```
('80389123000101',  4300, 201901)
('80389123000101',  4400, 201902)
('80389123000101',  4500, 201903)
('80389123000101',  4900, 201904)
('80389123000101',  4400, 201905)

('80389123000103',  1600, 201901)
('80389123000103',  1200, 201902)
('80389123000103',  1100, 201903)
('80389123000103',  1300, 201904)
('80389123000103',  1300, 201905)
```

Window Functions

```
SELECT *,
        AVG(vlr_transacao) OVER
(PARTITION BY pk_cd_cnpj ORDER BY date
ROWS BETWEEN 2 PRECEDING AND
CURRENT ROW) AS moving_avg
FROM t_employee;
```

```
('80389123000101',  4300, 201901)
('80389123000101',  4400, 201902)
('80389123000101',  4500, 201903)
('80389123000101',  4900, 201904)
('80389123000101',  4400, 201905)


('80389123000103',  1600, 201901)
('80389123000103',  1200, 201902)
('80389123000103',  1100, 201903)
('80389123000103',  1300, 201904)
('80389123000103',  1300, 201905)
```

Window Functions

```
SELECT *,
        AVG(vlr_transacao) OVER
(PARTITION BY pk_cd_cnpj ORDER BY date
ROWS BETWEEN 2 PRECEDING AND
CURRENT ROW) AS moving_avg
FROM t_employee;
```

```
('80389123000101',  4300, 201901)
('80389123000101',  4400, 201902)
('80389123000101',  4500, 201903)
('80389123000101',  4900, 201904)
('80389123000101',  4400, 201905)


('80389123000103',  1600, 201901)
('80389123000103',  1200, 201902)
('80389123000103',  1100, 201903)
('80389123000103',  1300, 201904)
('80389123000103',  1300, 201905)
```