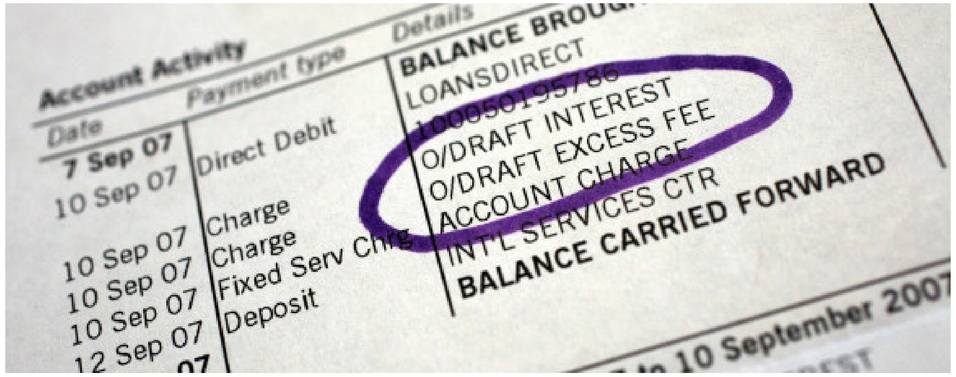
Hive PTF

Window or Analytics Functions


BANK


Account Activity
Date
Payment type
Details
7 Sep 07
10 Sep 07
Direct Debit
LOANSDIRECT
O/DRAFT INTEREST
O/DRAFT EXCESS FEE
ACCOUNT CHARGE
INT'L SERVICES CTR
BALANCE CARRIED FORWARD
10 Sep 07
10 Sep 07
10 Sep 07
12 Sep 07
Charge
Charge
Fixed Serv Chrg
Deposit


BANK


Account Activity
Date
Payment type
Details
7 Sep 07
10 Sep 07
Direct Debit
BALANCE BROUG
LOANSDIRECT
10005019578
10 Sep 07
Charge
O/DRAFT INTEREST
10 Sep 07
Charge
O/DRAFT EXCESS FEE
10 Sep 07
Fixed Serv Chrg
ACCOUNT CHARGE
12 Sep 07
Deposit
INT'L SERVICES CTR
BALANCE CARRIED FORWARD
10 September 2007

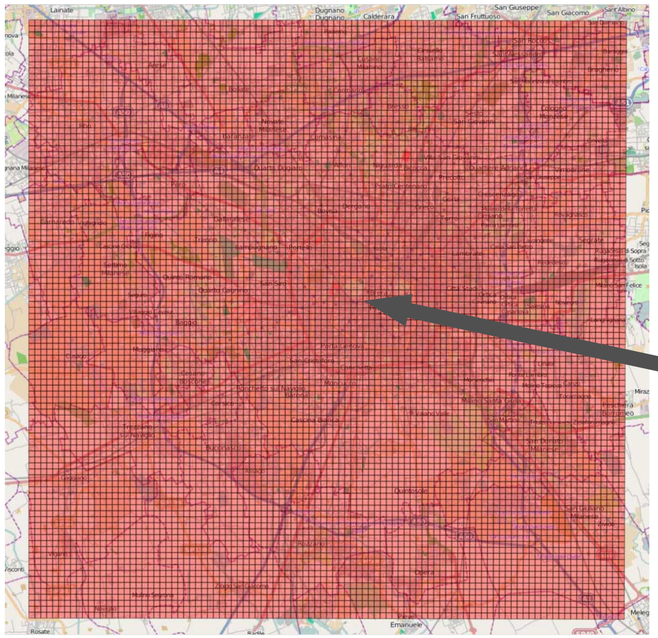
...


ATM


BANK


Account Activity
Date
Payment type
Details
7 Sep 07
10 Sep 07
Direct Debit
BALANCE BROUG
LOANSDIRECT
O/DRAFT INTEREST
O/DRAFT EXCESS FEE
ACCOUNT CHARGE
INT'L SERVICES CTR
BALANCE CARRIED FORWARD
10 Sep 07
10 Sep 07
10 Sep 07
12 Sep 07
Charge
Charge
Fixed Serv Chrg
Deposit
10 September 2007

...




ATM


```
FROM transactions
   SELECT TRANSFORM (customerID, change)
   USING "./locate_overdraft.py"
DISTRIBUTE BY customerID
SORT BY customerID, timestamp
```


```
FROM transactions
   SELECT TRANSFORM (customerID, change)
   USING "./locate_overdraft.py"
DISTRIBUTE BY customerID
SORT BY customerID, timestamp
```

Partitioned Table Functions (PTF)

1:1 1. Functions (UDFs = User Defined Functions)

n:1 2. Aggregate functions (UDAFs)

1:n 3. Table-generating functions (UDTFs)

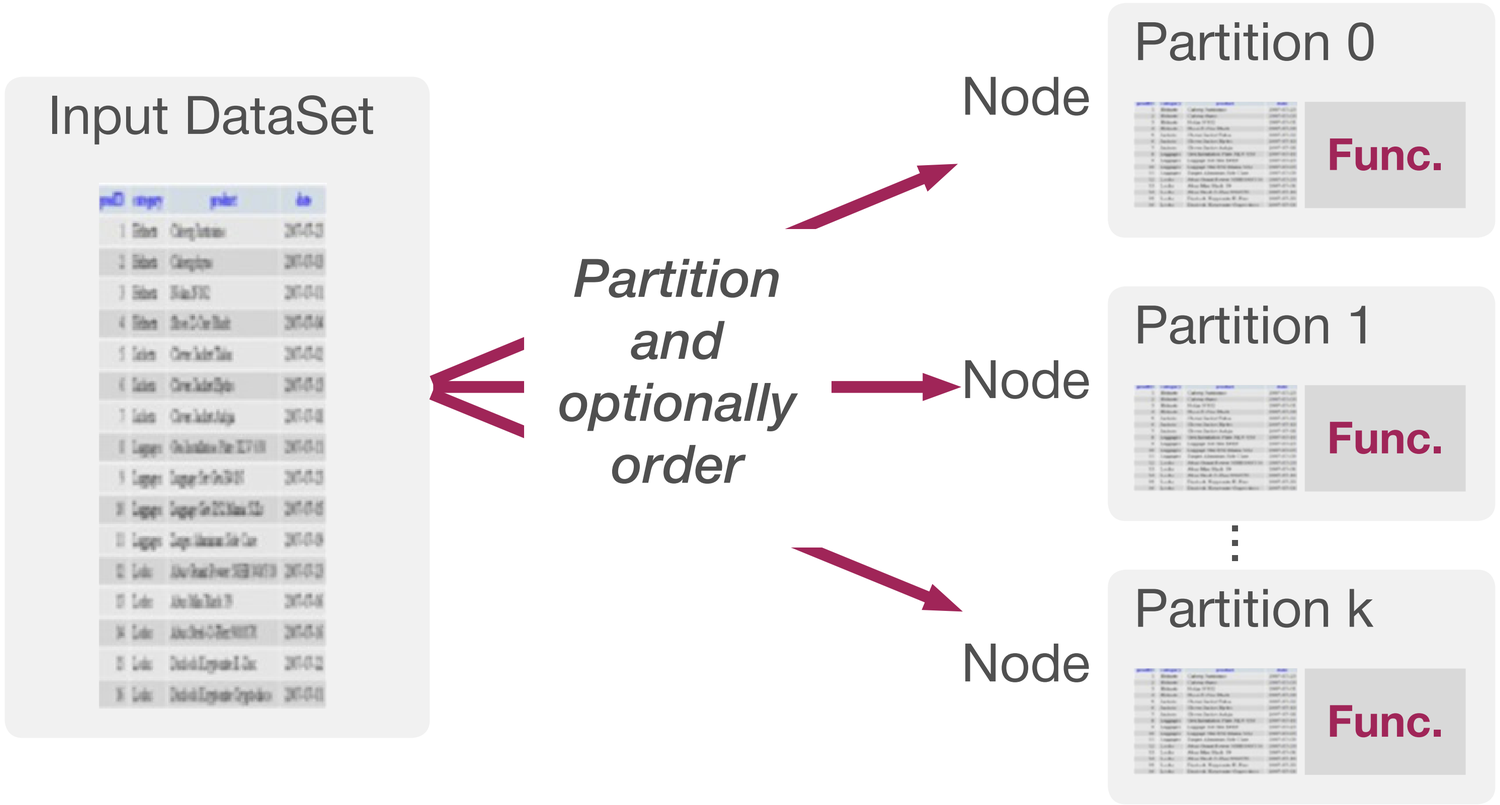
n:m 4. Partitioned table functions (PTFs)

1:1 1. Functions (UDFs = User Defined Functions)

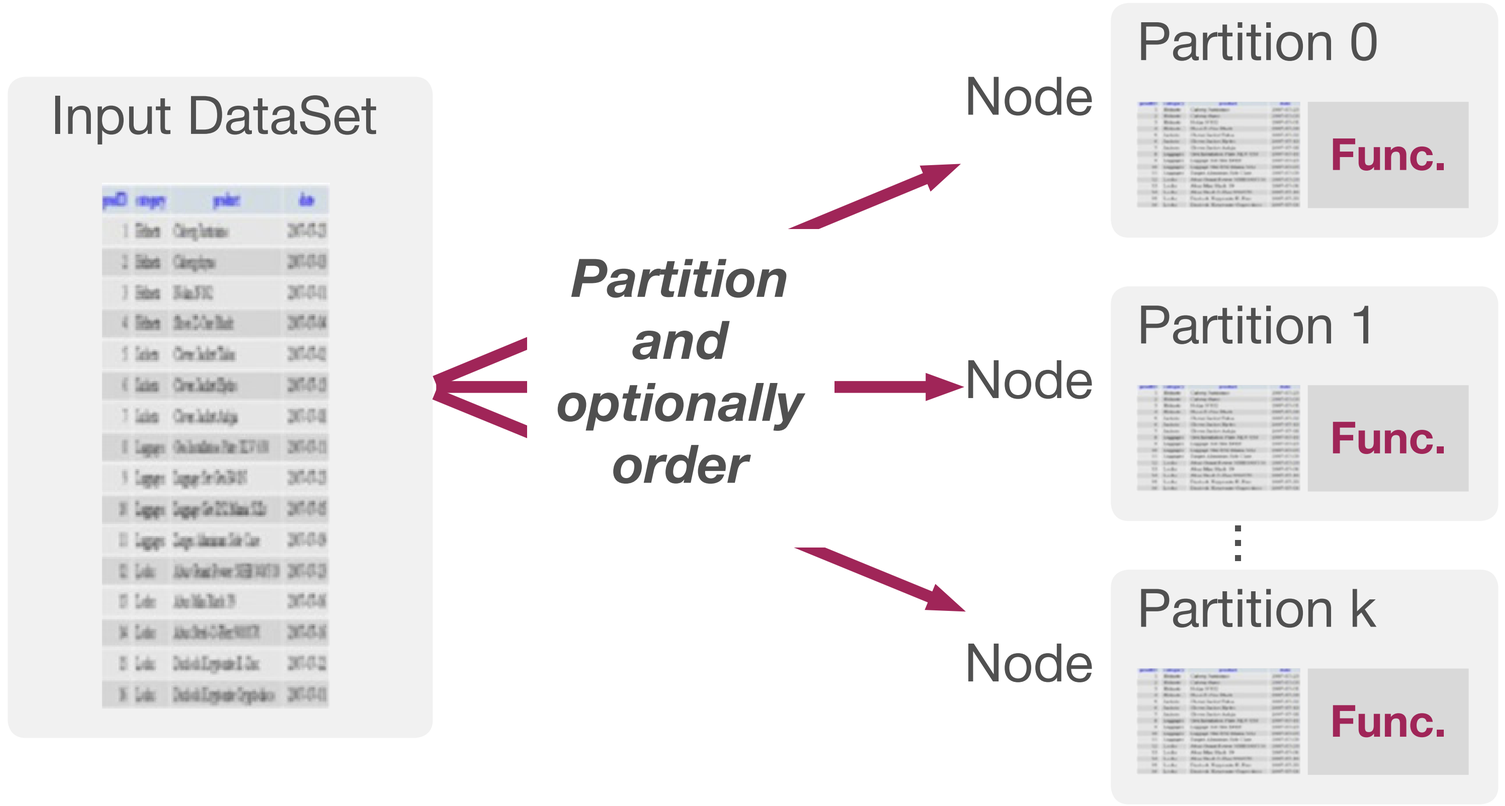
n:1 2. Aggregate functions (UDAFs)

1:n 3. Table-generating functions (UDTFs)

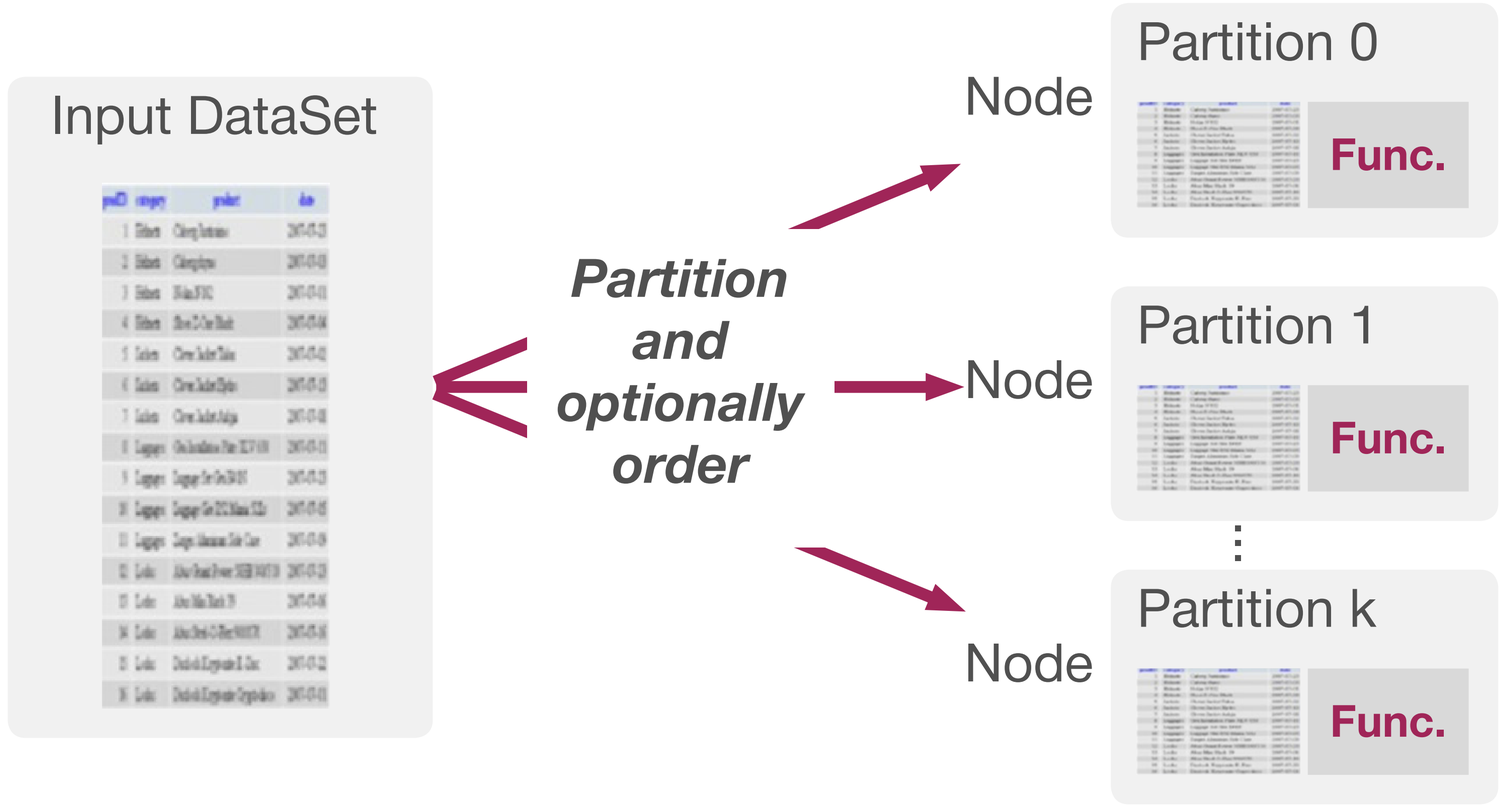
n:m 4. Partitioned table functions (PTFs)



```sql
SELECT column_A, ROW_NUMBER() OVER (PARTITION BY column_C)
FROM table_name;

SELECT column_A, RANK() OVER (PARTITION BY column_C)
FROM table_name;

SELECT column_A, DENSE_RANK() OVER (PARTITION BY column_C)
FROM table_name;
```



```sql
SELECT column_A, ROW_NUMBER() OVER (PARTITION BY column_C)
FROM table_name;

SELECT column_A, RANK() OVER (PARTITION BY column_C)
FROM table_name;

SELECT column_A, DENSE_RANK() OVER (PARTITION BY column_C)
FROM table_name;
```



```sql
SELECT column_A, ROW_NUMBER() OVER (PARTITION BY column_C)
FROM table_name;

SELECT column_A, RANK() OVER (PARTITION BY column_C)
FROM table_name;

SELECT column_A, DENSE_RANK() OVER (PARTITION BY column_C)
FROM table_name;
```

V	ROW_NUMBER
a	1
a	2
a	3
b	4
c	5
c	6
d	7
e	8

V	RANK
a	1
a	1
a	1
b	4
c	5
c	5
d	7
e	8

V	DENSE_RANK
a	1
a	1
a	1
b	2
c	3
c	3
d	4
e	5

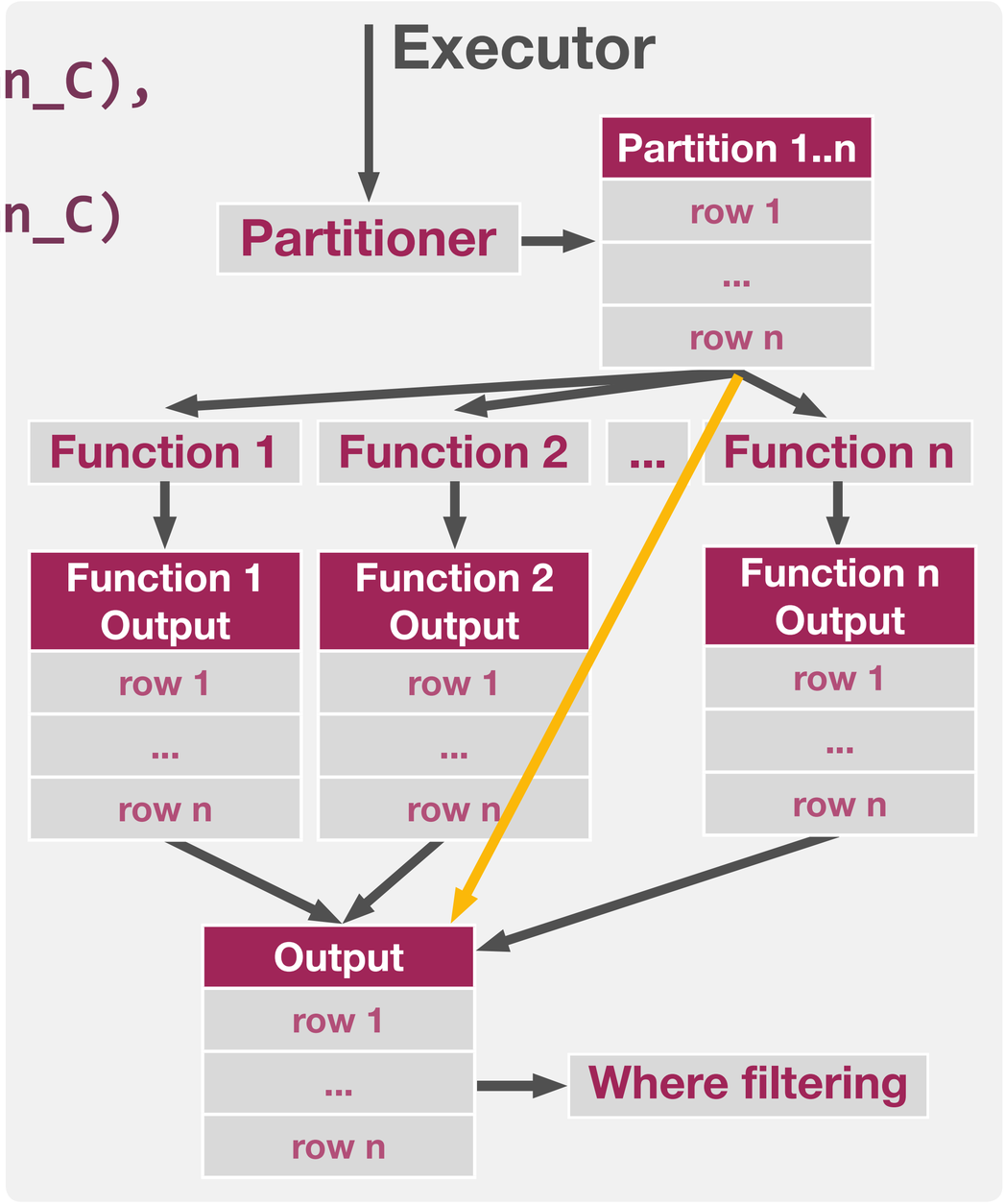
```sql
SELECT column_A,
    ROW_NUMBER() OVER (PARTITION BY column_C),
    RANK() OVER (PARTITION BY column_C),
    DENSE_RANK() OVER (PARTITION BY column_C)
FROM table_name;
```

```sql
SELECT column_A,
    ROW_NUMBER() OVER (PARTITION BY column_C),
    RANK() OVER (PARTITION BY column_C),
    DENSE_RANK() OVER (PARTITION BY column_C)
FROM table_name;
```

```sql
SELECT column_A,
    ROW_NUMBER() OVER w,
    RANK() OVER w,
    DENSE_RANK() OVER w
FROM table_name
WINDOW w AS (PARTITION BY column_C);
```

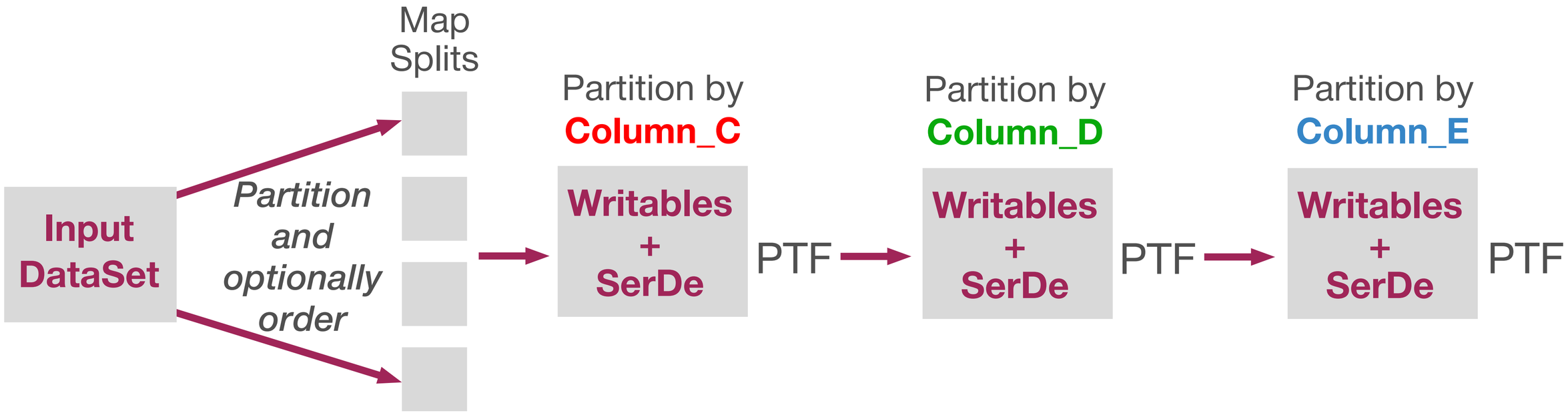
```sql
SELECT column_A,
    ROW_NUMBER() OVER (PARTITION BY column_C),
    RANK() OVER (PARTITION BY column_C),
    DENSE_RANK() OVER (PARTITION BY column_C)
FROM table_name;
```

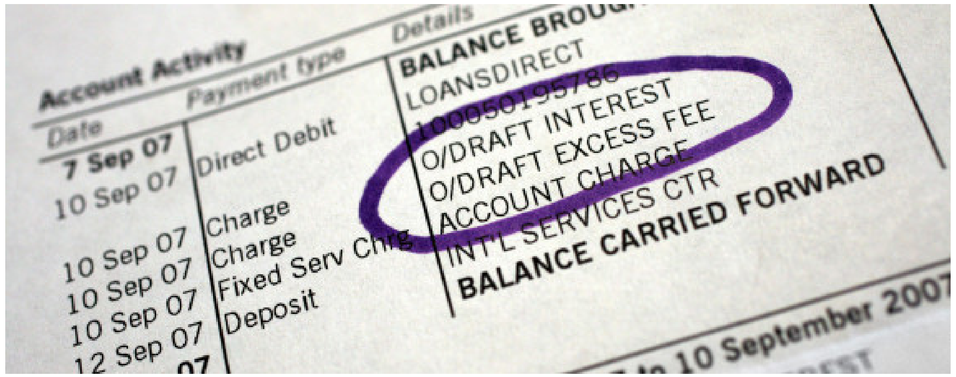
```sql
SELECT column_A,
    ROW_NUMBER() OVER w,
    RANK() OVER w,
    DENSE_RANK() OVER w
FROM table_name
WINDOW w AS (PARTITION BY column_C);
```



```sql
SELECT column_A,
    ROW_NUMBER() OVER (PARTITION BY column_C),
    RANK() OVER (PARTITION BY column_D),
    DENSE_RANK() OVER (PARTITION BY column_E)
FROM table_name;
```

```
SELECT column_A,
    ROW_NUMBER() OVER (PARTITION BY column_C),
    RANK() OVER (PARTITION BY column_D),
    DENSE_RANK() OVER (PARTITION BY column_E)
FROM table_name;
```






```sql
SELECT
    customerID,
    transactionID
    change,
    SUM(change) OVER (
      PARTITION BY customerID
      ORDER BY transactionID
    )
FROM transactions
SORT BY customerID, transactionID;
```

CustomerID	TransactionID	Change	Balance
42	1	1.00	1.00
42	2	-2.00	-1.00
42	3	10.00	9.00
42	4	-4.00	5.00
42	5	5.50	10.50

Summary

Summary

- You can **efficiently compute** "overdraft" for each customer using PTF

Summary

- You can **efficiently compute** "overdraft" for each customer using PTF
- You can **explain** how and when to use row_number, rank, dense_rank window functions

Summary

- You can **efficiently compute** "overdraft" for each customer using PTF
- You can **explain** how and when to use row_number, rank, dense_rank window functions
- You can **inspect** from HiveQL query how many MapReduce jobs are necessary to complete the job

Summary

- You can **efficiently compute** "overdraft" for each customer using PTF
- You can **explain** how and when to use row_number, rank, dense_rank window functions
- You can **inspect** from HiveQL query how many MapReduce jobs are necessary to complete the job

see: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+WindowingAndAnalytics