Identify and Resolve Data Skew in Spark ETL Job

(30 minutes read)

(Yusuf Arif, Sr. Data Architect, Big Data - usuf@amazon.com)

What is Data Skew?

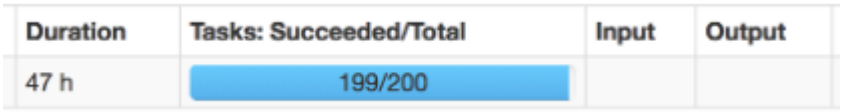
Data Skew is a condition in which a table's data is unevenly distributed among partitions across the nodes in the cluster.

During **Aggregation** and/or **Join** operations, **if data is unevenly distributed across partitions**; one or more tasks are stuck in processing the heavily skewed partition, resulting in **severely degraded performance** of queries, often the entire job fails due to massive shuffle. Joins between big tables require shuffling of data and skew can lead to an extreme **imbalance of work** in the cluster.

Joins and Aggregations are both shuffle operations, so the symptoms and fixes of Data Skew in both are the same.

Symptom of Data Skew

- A query appears to be **stuck**, finishing all but few (usually, the last 2 or 3 tasks out of default 200) tasks.
- A join stage seems to be **taking a long time** in Spark UI. This is typically one last task of the many tasks.
- Stages before and after join seem to be operating normally.
- Logs show "**Caused by: org.apache.spark.shuffle.FetchFailedException: Too large frame**:"



- On the Spark Web UI, you see something like:
- **Note:** 47 h in Duration can be any other value.

Potential Fixes of Data Skew

- Always ensure the **query is selecting only the relevant columns** working only on the data that you need for the join.
- Experiment with **different join orderings**, especially when the joins filter out a large amount of data.
- **Partition or repartition()** a dataset prior to joining for reducing data movement across the cluster, especially when the same dataset is used in multiple join operations. It's worth experimenting with different pre-join partitioning and repartitioning. **Note: All joins come at the cost of a shuffle**, this experimenting isn't free of cost but can be well worth in avoiding severely degraded performance later on.
- Ensure that the **null values are handled correctly** (using null, and not 'EMPTY' or empty space like " "). Given below are the **code examples to handle the null values in the joining columns**.

Code Example of Data Skew

Consider the following 3 tables, i) orders, ii) customers and iii) delivery. orders table has **300 million rows where cust_id is null** and **300 million rows where delivery_id is null**.

Since orders table has maximum data, it is the **driving** table, and should be the **left-most table** in any join operation.

Table Name	Row Count	Table Size	count(distinct cust_id)	cust_id is null	cust-id is not null
orders	500 million	50 GB	150 million	**300 million**	200 million
			count(distinct delivery_id)	**delivery_id is null**	**delivery-id is not null**
	500 million	50 GB	250 million	**300 million**	150 million
			count(distinct cust_id)	**cust_id is null**	**cust-id is not null**
customers	150 million	15 GB	150 million	0	150 million

			count(distinct delivery_id)	delivery_id is null	delivery-id is not null
delivery	250 million	25 GB	250 million	0	250 million

Sample Query joining 3 tables that results in Data Skew

```
select * from orders o
    left join customers c on o.cust_id = c.cust_id
    left join delivery d on o.dlivery_id = d.delivery_id;
```

In this query, marked in red are the problem areas. The bolded **red** is the **root-cause** of the Data Skew. orders table's **cust_id** and **delivery_id** h ave a **lot of null values** and are used as join-columns with customers table and delivery table.

Skewed Data

Since the driving table (orders) has null values in cust_id and delivery_id and we can't filter null records before joining due to some business requirement, we need all records from the driving table. When this query is run in a Spark, you will notice the stage performing this join progresses to 199 tasks quite fast and then gets stuck on the last task, eventually the query aborts after several hours and the job fails.

Reason for Data Skew

Spark **hashes the join columns and sorts it.** Then Spark tries to keep records with same hashes on the same executor, so all the null values from the orders table will go to one executor. Thus Spark gets into a continuous loop of shuffling and garbage collection with no success.

Fix the Data Skew

Split orders table into two parts. The first part will contain all rows that **don't have null** values for the **cust_id** and the second part contains all rows **with null values in cust_id**. Perform the **join between orders with no null values in cust_id and customers table**, then perform a **union all** of the two parts. Repeat the process for **delivery_id**.

The queries for splitting the orders table are:-

Query1:

```
create table orders_cust_id_not_null as
    select * from orders where cust_id is not null;
```

Query2:

```
create table orders_cust_id_null as
    select * from orders where cust_id is null;
```

Rewrite original Query:

```
create table orders_cust as
    (select o1.cust_id from orders_cust_id_not_null o1
        left join customer c on o1.cust_id = c.cust_id
    union all
    select o2.cust_id from orders_cust_id_null o2);
```

Repeat the process on delivery table

Query 3:

```
create table orders_delivery_id_not_null as
    select * from orders_cust where deliver_id is not null;
```

Query 4:

```
create table orders_delivery_id_null as
    select * from orders_cust where deliver_id is null;
```

Rewrite Original Query

```
create table  orders_delivery as
    (select o1.delivery_id from orders_delivery_id_not_null o1
        left join delivery d on o1.delivery_id = d.delivery_id
    union all
    select o2.delivery_id from orders_delivery_id_null o2);
```

Spark Code: Consider the 3 tables in csv format, given below.

orders table						**customers table**					**delivery table**			
orderd_id	**cust_id**	**delivery_id**	**order_dt**	**order_amt**		**cust_id**	**cust_name**	**cust_city**	**cust_state**		**delivery_id**	**delivery_dt**	**delivered_by**	**received_by**
1A	1	1001	1/1/18	100		1	Adam	San Francisco	CA		1001	1/10/18	Jon	YA
2A		1002	2/2/18	200		2	Bob	Palo Alto	CA		1002	2/10/18	Joe	MA
3A	2	1003	1/1/18	260		30	Charly	New York	NY		1003	1/10/81	Jack	HA
4A	3		3/3/18	300		4	Dave	San Antonio	TX		1004	3/10/18	Jared	SA
5A			4/4/18	150		5	Ed	Phoenix	AZ		1005	10-Apr	Jack	WA
6A	4	1004	5/5/18	400		6	Frank	Houston	TX		1006	5/10/18	Javed	SA
7A		1005	6/6/18	500		7	Gary	Madison	WI		1007	6/10/18	Juzar	SA

orders table does have null values in **cust_id** for order_id 2A, 5A and 7A and null values in **delivery_id** for order_id 4A and 5A. These null values are the reason why these columns are prone to **data skew**.

We will read these tables as csv files into DataFrames, and evaluate **how to resolve the data skew**.

```
orders = spark.read.csv("/Users/usuf/Documents/Customers/AmFm/orders.csv", header=True, inferSchema=True)
orders.show()
```

```
+---------+-------+-----------+--------+---------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|
+---------+-------+-----------+--------+---------+
|       1A|      1|       1001|  1/1/18|      100|
|       2A|   null|       1002|  2/2/18|      200|
|       3A|      2|       1003|  1/1/18|      260|
|       4A|      3|       null|  3/3/18|      300|
|       5A|   null|       null|  4/4/18|      150|
|       6A|      4|       1004|  5/5/18|      400|
|       7A|   null|       1005|  6/6/18|      500|
+---------+-------+-----------+--------+---------+
```

```
customers = spark.read.csv("/Users/usuf/Documents/Customers/AmFm/customers.csv", header=True, inferSchema=True)
customers.show()
```

```
+-------+---------+-------------+----------+
|cust_id|cust_name|    cust_city|cust_state|
+-------+---------+-------------+----------+
|      1|     Adam|San Francisco|        CA|
|      2|      Bob|    Palo Alto|        CA|
|     30|   Charly|     New York|        NY|
|      4|     Dave|  San Antonio|        TX|
|      5|       Ed|      Phoenix|        AZ|
|      6|    Frank|      Houston|        TX|
|      7|     Gary|      Madison|        WI|
+-------+---------+-------------+----------+
```

```
delivery = spark.read.csv("/Users/usuf/Documents/Customers/AmFm/delivery.csv", header=True, inferSchema=True)
delivery.show()
```

```
+-----------+-----------+------------+-----------+
|delivery_id|delivery_dt|delivered_by|received_by|
+-----------+-----------+------------+-----------+
|       1001|    1/10/18|         Jon|         YA|
|       1002|    2/10/18|         Joe|         MA|
|       1003|    1/10/81|        Jack|         HA|
|       1004|    3/10/18|       Jared|         SA|
|       1005|     10-Apr|        Jack|         WA|
|       1006|    5/10/18|       Javed|         SA|
|       1007|    6/10/18|       Juzar|         SA|
+-----------+-----------+------------+-----------+
```

To perform join in Spark, you can directly do so on DataFrame, or use **alias** of the DataFrame for easier access to the column names. Below are examples of both the techniques.

```
a_ord = orders.alias('a_ord')
a_cust = customers.alias('a_cust')
a_del = delivery.alias('a_del')
```

```
ocd = a_ord.join(a_cust, a_ord.cust_id == a_cust.cust_id, how='left_outer') \
           .join(a_del, a_ord.delivery_id == a_del.delivery_id, how='left_outer')
ocd.show()
```

```
+---------+-------+-----------+--------+---------+-------+---------+-------------+----------+-----------+-----------+
------------+-----------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_id|cust_name|    cust_city|cust_state|delivery_id|delivery_dt|
delivered_by|received_by|
+---------+-------+-----------+--------+---------+-------+---------+-------------+----------+-----------+-----------+
------------+-----------+
|       1A|      1|       1001|  1/1/18|      100|      1|     Adam|San Francisco|        CA|       1001|    1/10/18|
Jon|          YA|
|       2A|   null|       1002|  2/2/18|      200|   null|     null|         null|      null|       1002|    2/10/18|
Joe|          MA|
|       3A|      2|       1003|  1/1/18|      260|      2|      Bob|    Palo Alto|        CA|       1003|    1/10/81|
Jack|          HA|
|       4A|      3|       null|  3/3/18|      300|   null|     null|         null|      null|       null|       null|
null|        null|
|       5A|   null|       null|  4/4/18|      150|   null|     null|         null|      null|       null|       null|
null|        null|
|       6A|      4|       1004|  5/5/18|      400|      4|     Dave|  San Antonio|        TX|       1004|    3/10/18|
Jared|          SA|
|       7A|   null|       1005|  6/6/18|      500|   null|     null|         null|      null|       1005|     10-Apr|
Jack|          WA|
+---------+-------+-----------+--------+---------+-------+---------+-------------+----------+-----------+-----------+
------------+-----------+
```

The join in above code snippet is the **root cause of the data skew**, as it is joining on the **orders.cust_id** and **orders.delivery_id** columns, which have a lot of null values. Now, let's look into how to resolve

data skew issue in the multi table join where null values are present.

Resolving Data Skew

Task 1: **Split orders data_frame** into two data_frames, one where **cust_id is not null,** and the other where **cust_id is null**.

```
from pyspark.sql.functions import col
o_c_not_null = orders.filter(col("cust_id").isNotNull())
o_c_not_null.show()
```

```
+---------+-------+-----------+--------+---------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|
+---------+-------+-----------+--------+---------+
|       1A|      1|       1001|  1/1/18|      100|
|       3A|      2|       1003|  1/1/18|      260|
|       4A|      3|       null|  3/3/18|      300|
|       6A|      4|       1004|  5/5/18|      400|
+---------+-------+-----------+--------+---------+
```

```
o_c_null = orders.filter(col("cust_id").isNull())
o_c_null.show()
```

```
+---------+-------+-----------+--------+---------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|
+---------+-------+-----------+--------+---------+
|       2A|   null|       1002|  2/2/18|      200|
|       5A|   null|       null|  4/4/18|      150|
|       7A|   null|       1005|  6/6/18|      500|
+---------+-------+-----------+--------+---------+
```

Task 2: **Join o_c_not_null** DataFrame with **customers** DataFrame on **cust_id**. **Note:** This join will be much **faster**, as only the **non-null cust_ids are participating in the join**.

```
o_c_not_null_cust = o_c_not_null.join(customers, o_c_not_null['cust_id'] == customers['cust_id'], how='left_outer')
o_c_not_null_cust.show()

#select o1.cust_id from orders_cust_id_not_null o1
#left join customer c on o1.cust_id = c.cust_id
```

```
+---------+-------+-----------+--------+---------+-------+---------+-------------+----------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_id|cust_name|    cust_city|cust_state|
+---------+-------+-----------+--------+---------+-------+---------+-------------+----------+
|       1A|      1|       1001|  1/1/18|      100|      1|     Adam|San Francisco|        CA|
|       3A|      2|       1003|  1/1/18|      260|      2|      Bob|    Palo Alto|        CA|
|       4A|      3|       null|  3/3/18|      300|   null|     null|         null|      null|
|       6A|      4|       1004|  5/5/18|      400|      4|     Dave|  San Antonio|        TX|
+---------+-------+-----------+--------+---------+-------+---------+-------------+----------+
```

Problem: The cust_id column appears twice in the joined data_frame, as the column name is same in both the tables. If we try to access cust_id from the result data_frame, it will give 'AnalysisException' on cust_id due to the ambiguity.

```
AnalysisException: "Reference 'cust_id' is ambiguous, could be: cust_id, cust_id.;"
```

Sub_Task 2.1: **Remove duplicate columns** from the resulting data_frame. This is a good opportunity to introduce a User Defined Function, so it's generic on any two data_frames which may have duplicate columns.

```
def remove_duplicate_cols(df, df1, df2):
    repeated_columns = [c for c in df1.columns if c in df2.columns]
    for col in repeated_columns:
        df = df.drop(df2[col])
    return df
```

```
o_c_not_null_cust = remove_duplicate_cols(o_c_not_null_cust, o_c_not_null, customers)
o_c_not_null_cust.show()
```

```
+---------+-------+-----------+--------+---------+---------+-------------+----------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_name|    cust_city|cust_state|
+---------+-------+-----------+--------+---------+---------+-------------+----------+
|       1A|      1|       1001|  1/1/18|      100|     Adam|San Francisco|        CA|
|       3A|      2|       1003|  1/1/18|      260|      Bob|    Palo Alto|        CA|
|       4A|      3|       null|  3/3/18|      300|     null|         null|      null|
|       6A|      4|       1004|  5/5/18|      400|     Dave|  San Antonio|        TX|
+---------+-------+-----------+--------+---------+---------+-------------+----------+
```

Now we have the data_frame without the duplicate columns of cust_id.

Task 3: **Union** the **resulting** data_frame with **o_c_null** data_frame to get the full **orders_customers** data_frame.

In order to perform the union of the two data_frames, the **pre-requisite** is that both the **data_frames must have the same schema**. In this case, we need to union **o_c_not_null_cust** data_frame which is a joined product (with **customers** data_frame) and the **o_c_null** data_frame which is the sub-set of the orders data_frame. To do this, we have to customize the union, first ensuring the schemas are same in both the data_frames. This can be done with a generic User Defined Function.

```
from pyspark.sql.functions import lit
from pyspark.sql import Row

def customUnion(df1, df2):
    cols1 = df1.columns
    cols2 = df2.columns
    #total_cols = sorted(cols1 + list(set(cols2) - set(cols1))) #if columns need to be in sorted order
    total_cols = cols1 + list(set(cols2) - set(cols1))
    def expr(mycols, allcols):
        def processCols(colname):
            if colname in mycols:
                return colname
            else:
                return lit(None).alias(colname)
        cols = map(processCols, allcols)
        return list(cols)
    appended = df1.select(expr(cols1, total_cols)).union(df2.select(expr(cols2, total_cols)))
    return appended
```

```
ord_cust = customUnion(o_c_not_null_cust, o_c_null)
ord_cust.show()
```

```
+---------+-------+-----------+--------+---------+---------+-------------+----------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_name|    cust_city|cust_state|
+---------+-------+-----------+--------+---------+---------+-------------+----------+
|       1A|      1|       1001|  1/1/18|      100|     Adam|San Francisco|        CA|
|       3A|      2|       1003|  1/1/18|      260|      Bob|    Palo Alto|        CA|
|       4A|      3|       null|  3/3/18|      300|     null|         null|      null|
|       6A|      4|       1004|  5/5/18|      400|     Dave|  San Antonio|        TX|
|       2A|   null|       1002|  2/2/18|      200|     null|         null|      null|
|       5A|   null|       null|  4/4/18|      150|     null|         null|      null|
|       7A|   null|       1005|  6/6/18|      500|     null|         null|      null|
+---------+-------+-----------+--------+---------+---------+-------------+----------+
```

```
ord_cust = customUnion(o_c_not_null_cust, o_c_null).sort(['orderd_id'])
ord_cust.show()
```

```
+---------+-------+-----------+--------+---------+---------+-------------+----------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_name|    cust_city|cust_state|
+---------+-------+-----------+--------+---------+---------+-------------+----------+
|       1A|      1|       1001|  1/1/18|      100|     Adam|San Francisco|        CA|
|       2A|   null|       1002|  2/2/18|      200|     null|         null|      null|
|       3A|      2|       1003|  1/1/18|      260|      Bob|    Palo Alto|        CA|
|       4A|      3|       null|  3/3/18|      300|     null|         null|      null|
|       5A|   null|       null|  4/4/18|      150|     null|         null|      null|
|       6A|      4|       1004|  5/5/18|      400|     Dave|  San Antonio|        TX|
|       7A|   null|       1005|  6/6/18|      500|     null|         null|      null|
+---------+-------+-----------+--------+---------+---------+-------------+----------+
```

Now, that we have the **ord_cust** data_frame, we need to **perform the join with delivery table**. We will repeat the same steps, using the **ord_cust** data_frame as the **driving table**.

Task 4: **Split ord_cust** DataFrame into two DataFrames, one with **delivery_id not null** and the other with **deliver_id null**.

```
o_d_not_null = ord_cust.filter(col("delivery_id").isNotNull())
o_d_not_null.show()
```

```
+---------+-------+-----------+--------+---------+---------+-------------+----------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_name|    cust_city|cust_state|
+---------+-------+-----------+--------+---------+---------+-------------+----------+
|       1A|      1|       1001|  1/1/18|      100|     Adam|San Francisco|        CA|
|       2A|   null|       1002|  2/2/18|      200|     null|         null|      null|
|       3A|      2|       1003|  1/1/18|      260|      Bob|    Palo Alto|        CA|
|       6A|      4|       1004|  5/5/18|      400|     Dave|  San Antonio|        TX|
|       7A|   null|       1005|  6/6/18|      500|     null|         null|      null|
+---------+-------+-----------+--------+---------+---------+-------------+----------+
```

```
o_d_null = ord_cust.filter(col("delivery_id").isNull())
o_d_null.show()
```

```
+---------+-------+-----------+--------+---------+---------+---------+----------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_name|cust_city|cust_state|
+---------+-------+-----------+--------+---------+---------+---------+----------+
|       4A|      3|       null|  3/3/18|      300|     null|     null|      null|
|       5A|   null|       null|  4/4/18|      150|     null|     null|      null|
+---------+-------+-----------+--------+---------+---------+---------+----------+
```

Task 5: **Join** the **o_d_not_null** DataFrame with **delivery** DataFrame. This join will be much **faster**, **sans all the null values in delivery_id** column.

```
o_d_not_null_del = o_d_not_null.join(delivery, o_d_not_null['delivery_id'] == delivery['delivery_id'], how='left_outer'
o_d_not_null_del.show()
```

```
+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+-----------+--------
----+-----------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_name|    cust_city|cust_state|delivery_id|delivery_dt|delivere
d_by|received_by|
+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+-----------+--------
----+-----------+
|       1A|      1|       1001|  1/1/18|      100|     Adam|San Francisco|        CA|       1001|    1/10/18|
Jon|         YA|
|       2A|   null|       1002|  2/2/18|      200|     null|         null|      null|       1002|    2/10/18|
Joe|         MA|
|       3A|      2|       1003|  1/1/18|      260|      Bob|    Palo Alto|        CA|       1003|    1/10/81|
Jack|         HA|
|       6A|      4|       1004|  5/5/18|      400|     Dave|  San Antonio|        TX|       1004|    3/10/18|       J
ared|         SA|
|       7A|   null|       1005|  6/6/18|      500|     null|         null|      null|       1005|     10-Apr|
Jack|         WA|
+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+-----------+--------
----+-----------+
```

Task 6: **Drop duplicate column** from the resulting data_frame and **union** it with **o_d_null** data_frame re-using both the UDFs created earlier. The result of the union is the final data_frame, with all the 3 tables joined.

```
o_d_not_null_del = remove_duplicate_cols(o_d_not_null_del, o_d_not_null, delivery)
o_d_not_null_del.show()
```

```
+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+------------+-------
----+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_name|    cust_city|cust_state|delivery_dt|delivered_by|receive
d_by|
+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+------------+-------
----+
|       1A|      1|       1001|  1/1/18|      100|     Adam|San Francisco|        CA|    1/10/18|         Jon|
YA|
|       2A|   null|       1002|  2/2/18|      200|     null|         null|      null|    2/10/18|         Joe|
MA|
|       3A|      2|       1003|  1/1/18|      260|      Bob|    Palo Alto|        CA|    1/10/81|        Jack|
HA|
|       6A|      4|       1004|  5/5/18|      400|     Dave|  San Antonio|        TX|    3/10/18|       Jared|
SA|
|       7A|   null|       1005|  6/6/18|      500|     null|         null|      null|     10-Apr|        Jack|
WA|
+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+------------+-------
----+
```

```
ord_cust_del = customUnion(o_d_not_null_del, o_d_null).sort(['orderd_id'])
ord_cust_del.show()
```

```
+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+------------+-------
----+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_name|    cust_city|cust_state|delivery_dt|delivered_by|receive
d_by|
+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+------------+-------
----+
|       1A|      1|       1001|  1/1/18|      100|     Adam|San Francisco|        CA|    1/10/18|         Jon|
YA|
|       2A|   null|       1002|  2/2/18|      200|     null|         null|      null|    2/10/18|         Joe|
MA|
|       3A|      2|       1003|  1/1/18|      260|      Bob|    Palo Alto|        CA|    1/10/81|        Jack|
HA|
|       4A|      3|       null|  3/3/18|      300|     null|         null|      null|       null|        null|
null|
|       5A|   null|       null|  4/4/18|      150|     null|         null|      null|       null|        null|
null|
|       6A|      4|       1004|  5/5/18|      400|     Dave|  San Antonio|        TX|    3/10/18|       Jared|
SA|
|       7A|   null|       1005|  6/6/18|      500|     null|         null|      null|     10-Apr|        Jack|
WA|
+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+------------+-------
----+
```

In this way, you can **avoid tons of shuffle between the nodes in the cluster**, **prevent memory pressure on the driver and executor nodes**, **safe-guard against the data skews when null values are present**, leading to a performant Spark ETL, potentially saving **$$$$**.

---

raw-code: Below is the raw code from PySpark's ipython notebook.

```
orders = spark.read.csv("/Users/usuf/Documents/Customers/AmFm/orders.csv", header=True, inferSchema=True)
orders.show()
```

```
+---------+-------+-----------+--------+---------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|
+---------+-------+-----------+--------+---------+
|       1A|      1|       1001|  1/1/18|      100|
|       2A|   null|       1002|  2/2/18|      200|
|       3A|      2|       1003|  1/1/18|      260|
|       4A|      3|       null|  3/3/18|      300|
|       5A|   null|       null|  4/4/18|      150|
|       6A|      4|       1004|  5/5/18|      400|
|       7A|   null|       1005|  6/6/18|      500|
+---------+-------+-----------+--------+---------+
```

```
customers = spark.read.csv("/Users/usuf/Documents/Customers/AmFm/customers.csv", header=True, inferSchema=T
rue)
customers.show()
```

```
+-------+---------+-------------+----------+
|cust_id|cust_name|    cust_city|cust_state|
+-------+---------+-------------+----------+
|      1|     Adam|San Francisco|        CA|
|      2|      Bob|    Palo Alto|        CA|
|     30|   Charly|     New York|        NY|
|      4|     Dave|  San Antonio|        TX|
|      5|       Ed|      Phoenix|        AZ|
|      6|    Frank|      Houston|        TX|
|      7|     Gary|      Madison|        WI|
+-------+---------+-------------+----------+
```

```
delivery = spark.read.csv("/Users/usuf/Documents/Customers/AmFm/delivery.csv", header=True, inferSchema=Tru
e)
delivery.show()
```

```
+-----------+-----------+------------+-----------+
|delivery_id|delivery_dt|delivered_by|received_by|
+-----------+-----------+------------+-----------+
|       1001|    1/10/18|         Jon|         YA|
|       1002|    2/10/18|         Joe|         MA|
|       1003|    1/10/81|        Jack|         HA|
|       1004|    3/10/18|       Jared|         SA|
|       1005|     10-Apr|        Jack|         WA|
|       1006|    5/10/18|       Javed|         SA|
|       1007|    6/10/18|       Juzar|         SA|
+-----------+-----------+------------+-----------+
```

```
a_ord = orders.alias('a_ord')
a_cust = customers.alias('a_cust')
a_del = delivery.alias('a_del')

ocd = a_ord.join(a_cust, a_ord.cust_id == a_cust.cust_id, how='left_outer') \
          .join(a_del, a_ord.delivery_id == a_del.delivery_id, how='left_outer')
ocd.show()
```

```
+---------+-------+-----------+--------+---------+-------+---------+-------------+----------+-----------+--
---------+------------+-----------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_id|cust_name|
cust_city|cust_state|delivery_id|delivery_dt|delivered_by|received_by|
+---------+-------+-----------+--------+---------+-------+---------+-------------+----------+-----------+--
---------+------------+-----------+
|       1A|      1|       1001|  1/1/18|      100|      1|     Adam|San Francisco|        CA|       1001|
1/10/18|         Jon|         YA|
|       2A|   null|       1002|  2/2/18|      200|   null|     null|         null|      null|       1002|
2/10/18|         Joe|         MA|
|       3A|      2|       1003|  1/1/18|      260|      2|      Bob|    Palo Alto|        CA|       1003|
1/10/81|        Jack|         HA|
|       4A|      3|       null|  3/3/18|      300|   null|     null|         null|      null|       null|
null|        null|       null|
|       5A|   null|       null|  4/4/18|      150|   null|     null|         null|      null|       null|
null|        null|       null|
|       6A|      4|       1004|  5/5/18|      400|      4|     Dave|  San Antonio|        TX|       1004|
3/10/18|       Jared|         SA|
|       7A|   null|       1005|  6/6/18|      500|   null|     null|         null|      null|       1005|
10-Apr|        Jack|         WA|
+---------+-------+-----------+--------+---------+-------+---------+-------------+----------+-----------+--
---------+------------+-----------+
```

```
from pyspark.sql.functions import col
o_c_not_null = orders.filter(col("cust_id").isNotNull())
o_c_not_null.show()
```

```
+---------+-------+-----------+--------+---------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|
+---------+-------+-----------+--------+---------+
|       1A|      1|       1001|  1/1/18|      100|
|       3A|      2|       1003|  1/1/18|      260|
|       4A|      3|       null|  3/3/18|      300|
|       6A|      4|       1004|  5/5/18|      400|
+---------+-------+-----------+--------+---------+
```

```
o_c_null = orders.filter(col("cust_id").isNull())
o_c_null.show()
```

```
+---------+-------+-----------+--------+---------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|
+---------+-------+-----------+--------+---------+
|       2A|   null|       1002|  2/2/18|      200|
|       5A|   null|       null|  4/4/18|      150|
|       7A|   null|       1005|  6/6/18|      500|
+---------+-------+-----------+--------+---------+
```

```
o_c_not_null_cust = o_c_not_null.join(customers, o_c_not_null['cust_id'] == customers['cust_id'], how='left
_outer')
o_c_not_null_cust.show()

#select o1.cust_id from orders_cust_id_not_null o1
#left join customer c on o1.cust_id = c.cust_id
```

```
+---------+-------+-----------+--------+---------+-------+---------+-------------+----------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_id|cust_name|    cust_city|cust_state|
+---------+-------+-----------+--------+---------+-------+---------+-------------+----------+
|       1A|      1|       1001|  1/1/18|      100|      1|     Adam|San Francisco|        CA|
|       3A|      2|       1003|  1/1/18|      260|      2|      Bob|    Palo Alto|        CA|
|       4A|      3|       null|  3/3/18|      300|   null|     null|         null|      null|
|       6A|      4|       1004|  5/5/18|      400|      4|     Dave|  San Antonio|        TX|
+---------+-------+-----------+--------+---------+-------+---------+-------------+----------+
```

```
data = [
    Row(zip_code=78258, dma='TX'),
    Row(zip_code=78149, dma='TX'),
    Row(zip_code=53704, dma='WI'),
    Row(zip_code=94538, dma='CA')
]

firstDF = spark.createDataFrame(data)

data = [
    Row(zip_code='782', name='TX'),
    Row(zip_code='781', name='TX'),
    Row(zip_code='537', name='WI'),
    Row(zip_code='945', name='CA')
]

secondDF = spark.createDataFrame(data)

customUnion(firstDF,secondDF).show()
```

```
+----+----+--------+
| dma|name|zip_code|
+----+----+--------+
|  TX|null|   78258|
|  TX|null|   78149|
|  WI|null|   53704|
|  CA|null|   94538|
|null|  TX|     782|
|null|  TX|     781|
|null|  WI|     537|
|null|  CA|     945|
+----+----+--------+
```

```
from pyspark.sql.functions import lit
from pyspark.sql import Row

def customUnion(df1, df2):
    cols1 = df1.columns
    cols2 = df2.columns
    #total_cols = sorted(cols1 + list(set(cols2) - set(cols1))) #if columns need to be in sorted order
    total_cols = cols1 + list(set(cols2) - set(cols1))
    def expr(mycols, allcols):
        def processCols(colname):
            if colname in mycols:
                return colname
            else:
                return lit(None).alias(colname)
        cols = map(processCols, allcols)
        return list(cols)
    appended = df1.select(expr(cols1, total_cols)).union(df2.select(expr(cols2, total_cols)))
    return appended

def remove_duplicate_cols(df, df1, df2):
    repeated_columns = [c for c in df1.columns if c in df2.columns]
    for col in repeated_columns:
        df = df.drop(df2[col])
    return df

o_c_not_null_cust = remove_duplicate_cols(o_c_not_null_cust, o_c_not_null, customers)
o_c_not_null_cust.show()
```

```
+---------+-------+-----------+--------+---------+---------+-------------+----------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_name|    cust_city|cust_state|
+---------+-------+-----------+--------+---------+---------+-------------+----------+
|       1A|      1|       1001|  1/1/18|      100|     Adam|San Francisco|        CA|
|       3A|      2|       1003|  1/1/18|      260|      Bob|    Palo Alto|        CA|
|       4A|      3|       null|  3/3/18|      300|     null|         null|      null|
|       6A|      4|       1004|  5/5/18|      400|     Dave|  San Antonio|        TX|
+---------+-------+-----------+--------+---------+---------+-------------+----------+
```

```
ord_cust = customUnion(o_c_not_null_cust, o_c_null).sort(['orderd_id'])
ord_cust.show()
```

```
+---------+-------+-----------+--------+---------+---------+-------------+----------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_name|    cust_city|cust_state|
+---------+-------+-----------+--------+---------+---------+-------------+----------+
|       1A|      1|       1001|  1/1/18|      100|     Adam|San Francisco|        CA|
|       2A|   null|       1002|  2/2/18|      200|     null|         null|      null|
|       3A|      2|       1003|  1/1/18|      260|      Bob|    Palo Alto|        CA|
|       4A|      3|       null|  3/3/18|      300|     null|         null|      null|
|       5A|   null|       null|  4/4/18|      150|     null|         null|      null|
|       6A|      4|       1004|  5/5/18|      400|     Dave|  San Antonio|        TX|
|       7A|   null|       1005|  6/6/18|      500|     null|         null|      null|
+---------+-------+-----------+--------+---------+---------+-------------+----------+
```

```
o_d_not_null = ord_cust.filter(col("delivery_id").isNotNull())
o_d_not_null.show()
```

```
+---------+-------+-----------+--------+---------+---------+-------------+----------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_name|    cust_city|cust_state|
+---------+-------+-----------+--------+---------+---------+-------------+----------+
|       1A|      1|       1001|  1/1/18|      100|     Adam|San Francisco|        CA|
|       2A|   null|       1002|  2/2/18|      200|     null|         null|      null|
|       3A|      2|       1003|  1/1/18|      260|      Bob|    Palo Alto|        CA|
|       6A|      4|       1004|  5/5/18|      400|     Dave|  San Antonio|        TX|
|       7A|   null|       1005|  6/6/18|      500|     null|         null|      null|
+---------+-------+-----------+--------+---------+---------+-------------+----------+
```

```
o_d_null = ord_cust.filter(col("delivery_id").isNull())
o_d_null.show()
```

```
+---------+-------+-----------+--------+---------+---------+---------+----------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_name|cust_city|cust_state|
+---------+-------+-----------+--------+---------+---------+---------+----------+
|       4A|      3|       null|  3/3/18|      300|     null|     null|      null|
|       5A|   null|       null|  4/4/18|      150|     null|     null|      null|
+---------+-------+-----------+--------+---------+---------+---------+----------+
```

```
o_d_not_null_del = o_d_not_null.join(delivery, o_d_not_null['delivery_id'] == delivery['delivery_id'], how=
'left_outer')
o_d_not_null_del.show()
```

```
+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+----------
-+------------+-----------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_name|
cust_city|cust_state|delivery_id|delivery_dt|delivered_by|received_by|
+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+----------
-+------------+-----------+
|       1A|      1|       1001|  1/1/18|      100|     Adam|San Francisco|        CA|       1001|
1/10/18|         Jon|         YA|
|       2A|   null|       1002|  2/2/18|      200|     null|         null|      null|       1002|
2/10/18|         Joe|         MA|
|       3A|      2|       1003|  1/1/18|      260|      Bob|    Palo Alto|        CA|       1003|
1/10/81|        Jack|         HA|
|       6A|      4|       1004|  5/5/18|      400|     Dave|  San Antonio|        TX|       1004|
3/10/18|       Jared|         SA|
|       7A|   null|       1005|  6/6/18|      500|     null|         null|      null|       1005|
10-Apr|        Jack|         WA|
+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+----------
-+------------+-----------+
```

```
o_d_not_null_del = remove_duplicate_cols(o_d_not_null_del, o_d_not_null, delivery)
o_d_not_null_del.show()
```

```
+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+----------
--+-----------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_name|
cust_city|cust_state|delivery_dt|delivered_by|received_by|
+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+----------
--+-----------+
|       1A|      1|       1001|  1/1/18|      100|     Adam|San Francisco|        CA|    1/10/18|
Jon|         YA|
|       2A|   null|       1002|  2/2/18|      200|     null|         null|      null|    2/10/18|
Joe|         MA|
|       3A|      2|       1003|  1/1/18|      260|      Bob|    Palo Alto|        CA|    1/10/81|
Jack|         HA|
|       6A|      4|       1004|  5/5/18|      400|     Dave|  San Antonio|        TX|    3/10/18|
Jared|         SA|
|       7A|   null|       1005|  6/6/18|      500|     null|         null|      null|     10-Apr|
Jack|         WA|
+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+----------
--+-----------+


ord_cust_del = customUnion(o_d_not_null_del, o_d_null).sort(['orderd_id'])
ord_cust_del.show()

+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+----------
--+-----------+
|orderd_id|cust_id|delivery_id|order_dt|order_amt|cust_name|
cust_city|cust_state|delivery_dt|delivered_by|received_by|
+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+----------
--+-----------+
|       1A|      1|       1001|  1/1/18|      100|     Adam|San Francisco|        CA|    1/10/18|
Jon|         YA|
|       2A|   null|       1002|  2/2/18|      200|     null|         null|      null|    2/10/18|
Joe|         MA|
|       3A|      2|       1003|  1/1/18|      260|      Bob|    Palo Alto|        CA|    1/10/81|
Jack|         HA|
|       4A|      3|       null|  3/3/18|      300|     null|         null|      null|       null|
null|       null|
|       5A|   null|       null|  4/4/18|      150|     null|         null|      null|       null|
null|       null|
|       6A|      4|       1004|  5/5/18|      400|     Dave|  San Antonio|        TX|    3/10/18|
Jared|         SA|
|       7A|   null|       1005|  6/6/18|      500|     null|         null|      null|     10-Apr|
Jack|         WA|
+---------+-------+-----------+--------+---------+---------+-------------+----------+-----------+----------
--+-----------+
```