



**Combine Two DF**

Write a Pyspark program to report the first name, last name, city, and state of each person in the Person dataframe. If the address of a personId is not present in the Address dataframe, report null instead.

**Difficult Level :** EASY

**DataFrame:**

**# Define schema for the 'persons' table**

**persons\_schema = StructType([**

**StructField("personId", IntegerType(), True),**

**StructField("lastName", StringType(), True),**

**StructField("firstName", StringType(), True)**

**])**

**# Define schema for the 'addresses' table**

**addresses\_schema = StructType([**

**StructField("addressId", IntegerType(), True),**

**StructField("personId", IntegerType(), True),**

**StructField("city", StringType(), True),**

**StructField("state", StringType(), True)**

**])**

**# Define data for the 'persons' table**

**persons\_data = [**

**(1, 'Wang', 'Allen'),**

**(2, 'Alice', 'Bob')**

**]**

**# Define data for the 'addresses' table**

**addresses\_data = [**

**(1, 2, 'New York City', 'New York'),**

**(2, 3, 'Leetcode', 'California')**

**]**



**INPUT**

| **INPUT-1 persons** | | |
| --- | --- | --- |
| **PERSONID** | **LASTNAME** | **FIRSTNAME** |
| **1** | **Wang** | **Allen** |
| **2** | **Alice** | **Bob** |

| **INPUT-2 addresses** | | | |
| --- | --- | --- | --- |
| **ADDRESSID** | **PERSONID** | **CITY** | **STATE** |
| **1** | **2** | **New York City** | **New York** |
| **2** | **3** | **Leetcode** | **California** |

**OUTPUT**

| **OUTPUT** | | | |
| --- | --- | --- | --- |
| **FIRSTNAME** | **LASTNAME** | **CITY** | **STATE** |
| **Bob** | **Alice** | **New York City** | **New York** |
| **Allen** | **Wang** |  |  |



**# Creating Spark Session**

**from pyspark.sql import SparkSession**

**from pyspark.sql.types import StructType,StructField,IntegerType,StringType**

**from pyspark.sql.functions import when**

**from pyspark.sql import functions as F**

**from pyspark.sql.window import Window**

**#creating spark session**

**spark = SparkSession. \**

**builder. \**

**config('spark.shuffle.useOldFetchProtocol', 'true'). \**

**config('spark.ui.port','0'). \**

**config("spark.sql.warehouse.dir", "/user/itv008042/warehouse"). \**

**enableHiveSupport(). \**

**master('yarn'). \**

**getOrCreate()**

**# Define schema for the 'persons' table**

**persons\_schema = StructType([**

**StructField("personId", IntegerType(), True),**

**StructField("lastName", StringType(), True),**

**StructField("firstName", StringType(), True)**

**])**

**# Define schema for the 'addresses' table**

**addresses\_schema = StructType([**

**StructField("addressId", IntegerType(), True),**

**StructField("personId", IntegerType(), True),**

**StructField("city", StringType(), True),**

**StructField("state", StringType(), True)**

**])**

**# Define data for the 'persons' table**

**persons\_data = [**

**(1, 'Wang', 'Allen'),**

**(2, 'Alice', 'Bob')**

**]**

**# Define data for the 'addresses' table**

**addresses\_data = [**

**(1, 2, 'New York City', 'New York'),**

**(2, 3, 'Leetcode', 'California')**

**]**

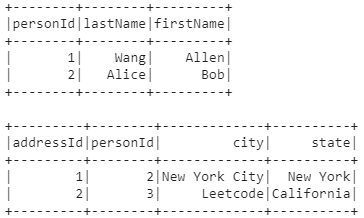
**# Create a PySpark DataFrame**

**person\_df=spark.createDataFrame(persons\_data,persons\_schema)**

**address\_df=spark.createDataFrame(addresses\_data,addresses\_schema)**

**person\_df.show()**

**address\_df.show()**

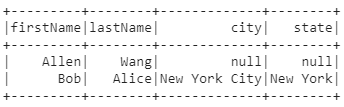
****

**person\_df.join(address\_df,person\_df.personId==address\_df.personId,'left')\**

**.select('firstName','lastName','city','state')\**

**.show()**

**# Show the result DataFrame**

****

