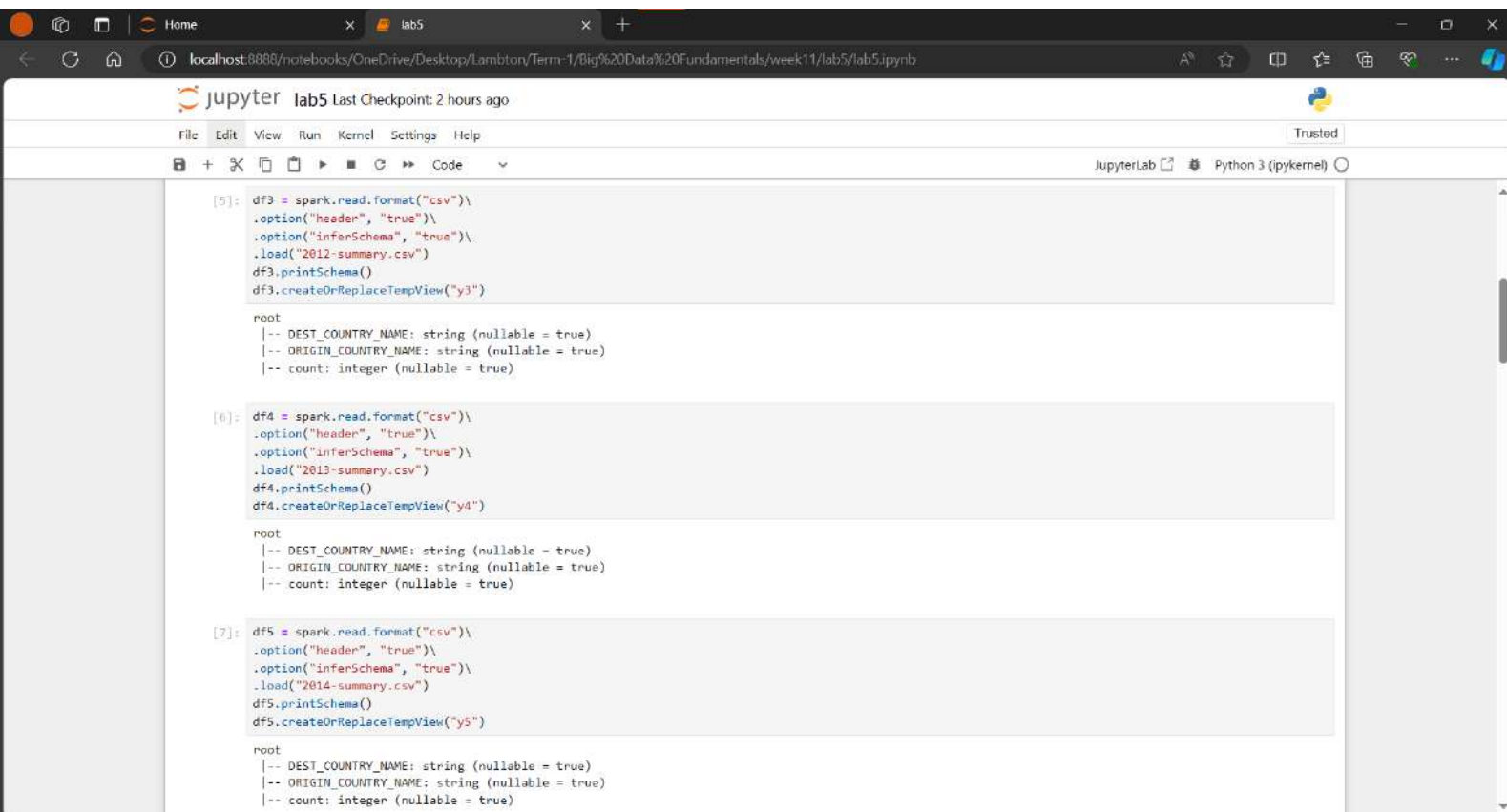


Name: Mugilmithran Kathiravan
Std ID: C0934419

LAB 5

1. Created six different dataframes using six csv files.



The screenshot shows a JupyterLab window titled 'lab5' with a 'Trusted' badge. The interface includes a menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar with icons for file operations and code execution. The main area contains three code cells, each loading a CSV file into a Spark DataFrame and displaying its schema.

```
[5]: df3 = spark.read.format("csv")\
      .option("header", "true")\
      .option("inferSchema", "true")\
      .load("2012-summary.csv")
df3.printSchema()
df3.createOrReplaceTempView("y3")

root
 |-- DEST_COUNTRY_NAME: string (nullable = true)
 |-- ORIGIN_COUNTRY_NAME: string (nullable = true)
 |-- count: integer (nullable = true)

[6]: df4 = spark.read.format("csv")\
      .option("header", "true")\
      .option("inferSchema", "true")\
      .load("2013-summary.csv")
df4.printSchema()
df4.createOrReplaceTempView("y4")

root
 |-- DEST_COUNTRY_NAME: string (nullable = true)
 |-- ORIGIN_COUNTRY_NAME: string (nullable = true)
 |-- count: integer (nullable = true)

[7]: df5 = spark.read.format("csv")\
      .option("header", "true")\
      .option("inferSchema", "true")\
      .load("2014-summary.csv")
df5.printSchema()
df5.createOrReplaceTempView("y5")

root
 |-- DEST_COUNTRY_NAME: string (nullable = true)
 |-- ORIGIN_COUNTRY_NAME: string (nullable = true)
 |-- count: integer (nullable = true)
```



```
[8]: df6 = spark.read.format("csv")\
    .option("header", "true")\
    .option("inferSchema", "true")\
    .load("2015-summary.csv")
df6.printSchema()
df6.createOrReplaceTempView("y6")

root
 |-- DEST_COUNTRY_NAME: string (nullable = true)
 |-- ORIGIN_COUNTRY_NAME: string (nullable = true)
 |-- count: integer (nullable = true)
```

2. Importing necessary modules and Concatenate all six DataFrames into one single DataFrame.

jupyter lab5 Last Checkpoint: 2 hours ago



File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

```
[66]: from functools import reduce
      from pyspark.sql import DataFrame
      from pyspark.sql.functions import lit, desc, col
```

```
[10]: #Concatenating multiple dataframes into a single dataframe.
```

```
df_list = [df1, df2, df3, df4, df5, df6]
```

```
df_all = reduce(DataFrame.unionAll, df_list)
```

```
#Adding year column to the table.
```

```
df_all = df_all.withColumn("year", lit(2010)).union(df2.withColumn("year", lit(2011))).union(df3.withColumn("year", lit(2012))).union(df4.withColumn("year", lit(2013))).union(df5.withColumn("year", lit(2014))).union(df6.withColumn("year", lit(2015)))
```

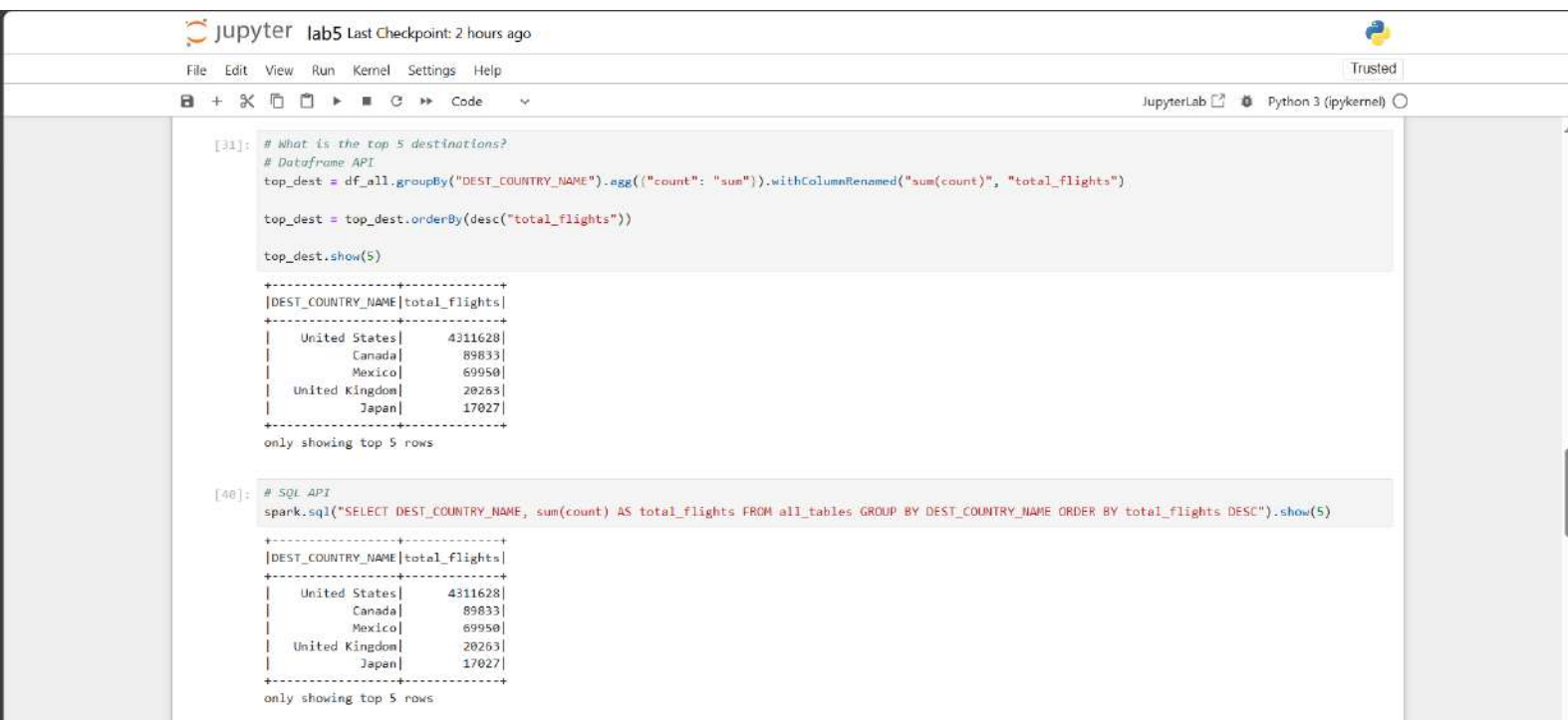
```
#Creating a table using the concatenated dataframe.
```

```
df_all.createOrReplaceTempView("all_tables")
```

```
[11]: spark.sql("SELECT * FROM all_tables").show()
```

DEST_COUNTRY_NAME	ORIGIN_COUNTRY_NAME	count	year
United States	Romania	1	2010
United States	Ireland	264	2010
United States	India	69	2010
Egypt	United States	24	2010
Equatorial Guinea	United States	1	2010
United States	Singapore	25	2010
United States	Grenada	54	2010
Costa Rica	United States	477	2010
Senegal	United States	29	2010
United States	Marshall Islands	44	2010
Guyana	United States	17	2010
United States	Sint Maarten	53	2010
Malta	United States	1	2010
Bolivia	United States	46	2010

3. Top 5 Destinations.



JupyterLab interface showing two code cells. The first cell uses the Dataframe API to find the top 5 destinations by total flights. The second cell uses the SQL API to achieve the same result. Both cells show the same output: a table with 5 rows of destination data.

```
[31]: # What is the top 5 destinations?
# Dataframe API
top_dest = df_all.groupBy("DEST_COUNTRY_NAME").agg({"count": "sum"}).withColumnRenamed("sum(count)", "total_flights")

top_dest = top_dest.orderBy(desc("total_flights"))

top_dest.show(5)
```

DEST_COUNTRY_NAME	total_flights
United States	4311628
Canada	89833
Mexico	69950
United Kingdom	20263
Japan	17027

only showing top 5 rows

```
[40]: # SQL API
spark.sql("SELECT DEST_COUNTRY_NAME, sum(count) AS total_flights FROM all_tables GROUP BY DEST_COUNTRY_NAME ORDER BY total_flights DESC").show(5)
```

DEST_COUNTRY_NAME	total_flights
United States	4311628
Canada	89833
Mexico	69950
United Kingdom	20263
Japan	17027

only showing top 5 rows

4. Top 5 busiest routes in 2014 and 2015 (combined)



```
[58]: # What is the top 5 busiest routes in 2014 and 2015 (combined)?
# Dataframe API
busiest_route = df_all.where((df_all["year"] == 2014) | (df_all["year"] == 2015)).groupBy("ORIGIN_COUNTRY_NAME", "DEST_COUNTRY_NAME").agg({"count": "sum"})
busiest_route = busiest_route.orderBy(desc("total_flights"))
busiest_route.show(5)
```

ORIGIN_COUNTRY_NAME	DEST_COUNTRY_NAME	total_flights
United States	United States	728356
Canada	United States	16660
United States	Canada	16373
Mexico	United States	13677
United States	Mexico	13567

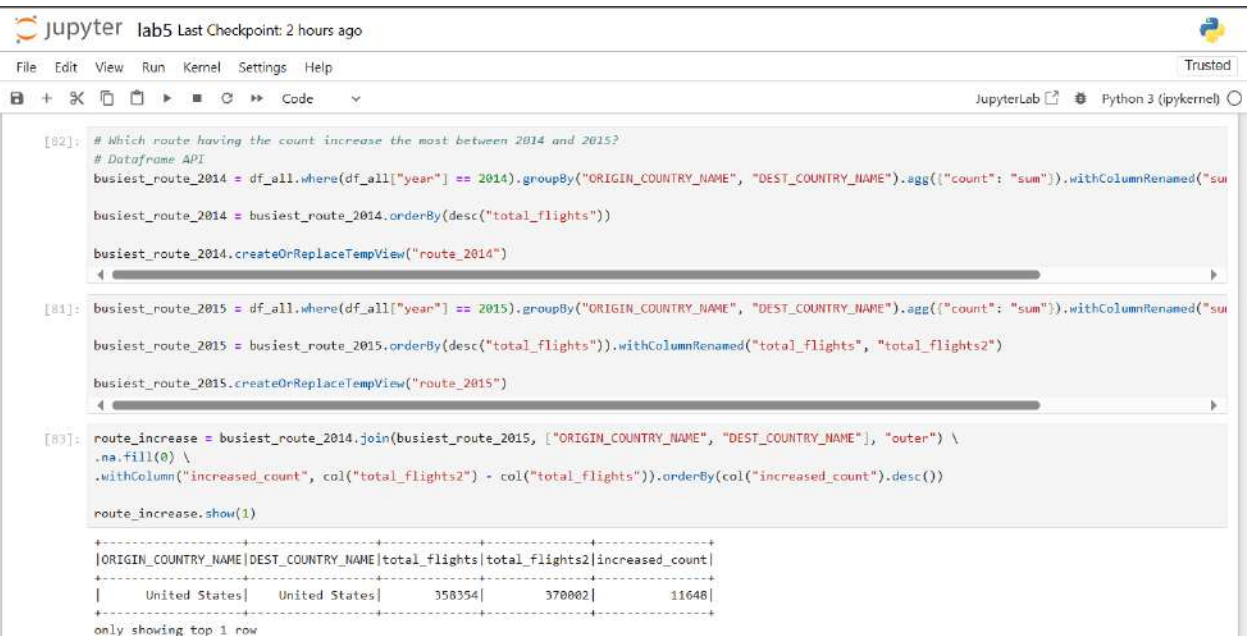
only showing top 5 rows

```
[46]: # SQL API
spark.sql("SELECT ORIGIN_COUNTRY_NAME, DEST_COUNTRY_NAME, sum(count) AS TOTAL_FLIGHTS FROM all_tables WHERE year = 2014 OR year = 2015 GROUP BY ORIGIN_COI")
```

ORIGIN_COUNTRY_NAME	DEST_COUNTRY_NAME	TOTAL_FLIGHTS
United States	United States	728356
Canada	United States	16660
United States	Canada	16373
Mexico	United States	13677
United States	Mexico	13567

only showing top 5 rows

5. Route having the count increase the most between 2014 and 2015



The image shows a JupyterLab interface with a Python notebook. The notebook contains three code cells. The first cell (index 80) filters the data for the year 2014, groups it by origin and destination country names, aggregates the total flights, and creates a temporary view named 'route_2014'. The second cell (index 81) does the same for the year 2015, creating a temporary view named 'route_2015'. The third cell (index 82) joins the two temporary views, calculates the increase in flight count, and displays the top row of the results.

```
[80]: # Which route having the count increase the most between 2014 and 2015?
# Dataframe API
busiest_route_2014 = df_all.where(df_all["year"] == 2014).groupBy("ORIGIN_COUNTRY_NAME", "DEST_COUNTRY_NAME").agg({"count": "sum").withColumnRenamed("sum", "total_flights")
busiest_route_2014 = busiest_route_2014.orderBy(desc("total_flights"))
busiest_route_2014.createOrReplaceTempView("route_2014")

[81]: busiest_route_2015 = df_all.where(df_all["year"] == 2015).groupBy("ORIGIN_COUNTRY_NAME", "DEST_COUNTRY_NAME").agg({"count": "sum").withColumnRenamed("sum", "total_flights2")
busiest_route_2015 = busiest_route_2015.orderBy(desc("total_flights2")).withColumnRenamed("total_flights2", "total_flights")
busiest_route_2015.createOrReplaceTempView("route_2015")

[82]: route_increased = busiest_route_2014.join(busiest_route_2015, ["ORIGIN_COUNTRY_NAME", "DEST_COUNTRY_NAME"], "outer") \
    .na.fill(0) \
    .withColumn("increased_count", col("total_flights2") - col("total_flights")).orderBy(col("increased_count").desc())
route_increased.show(1)
```

ORIGIN_COUNTRY_NAME	DEST_COUNTRY_NAME	total_flights	total_flights2	increased_count
United States	United States	358354	370002	11648

only showing top 1 row

```
[89]: # SQL API
spark.sql("""SELECT r1.ORIGIN_COUNTRY_NAME, r1.DEST_COUNTRY_NAME, (r2.total_flights2 - r1.total_flights) AS increased_count
FROM route_2014 r1
JOIN route_2015 r2
ON r1.ORIGIN_COUNTRY_NAME = r2.ORIGIN_COUNTRY_NAME
AND r1.DEST_COUNTRY_NAME = r2.DEST_COUNTRY_NAME
ORDER BY increased_count DESC
LIMIT 1""").show()
```

ORIGIN_COUNTRY_NAME	DEST_COUNTRY_NAME	increased_count
United States	United States	11648

```
[1]: from pyspark.sql import SparkSession

[2]: # Creating spark session
     spark = SparkSession.builder.getOrCreate()

[3]: # Loading 'csv' file data into spark dataframe.
     df1 = spark.read.format("csv")\
         .option("header", "true")\
         .option("inferSchema", "true")\
         .load("2010-summary.csv")
     df1.printSchema()
     df1.createOrReplaceTempView("y1")

root
 |-- DEST_COUNTRY_NAME: string (nullable = true)
 |-- ORIGIN_COUNTRY_NAME: string (nullable = true)
 |-- count: integer (nullable = true)

[4]: df2 = spark.read.format("csv")\
     .option("header", "true")\
     .option("inferSchema", "true")\
     .load("2011-summary.csv")
     df2.printSchema()
     df2.createOrReplaceTempView("y2")

root
 |-- DEST_COUNTRY_NAME: string (nullable = true)
 |-- ORIGIN_COUNTRY_NAME: string (nullable = true)
 |-- count: integer (nullable = true)
```