

Databricks Assignment

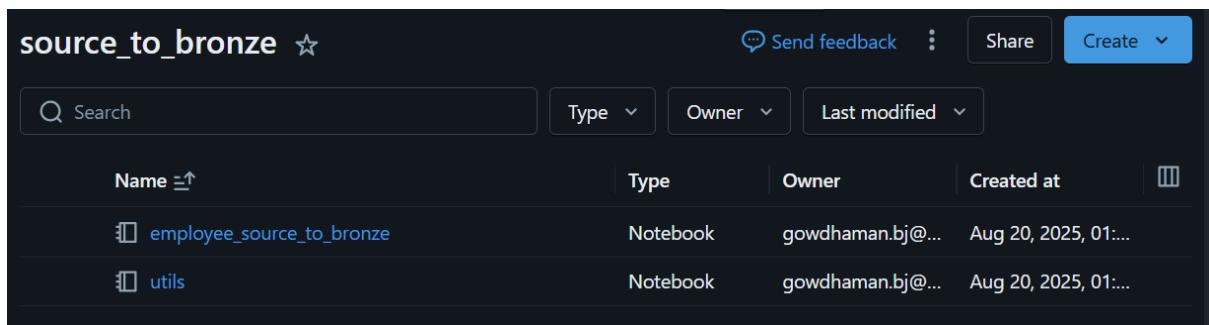
Question 1:

1. Create 3 folders as source_to_bronze, bronze_to_silver, silver_to_gold.

Name	Type	Owner	Created at
bronze_to_silver	Folder	gowdhaman.bj@...	Aug 20, 2025, 01:...
silver_to_gold	Folder	gowdhaman.bj@...	Aug 20, 2025, 01:...
source_to_bronze	Folder	gowdhaman.bj@...	Aug 20, 2025, 01:...

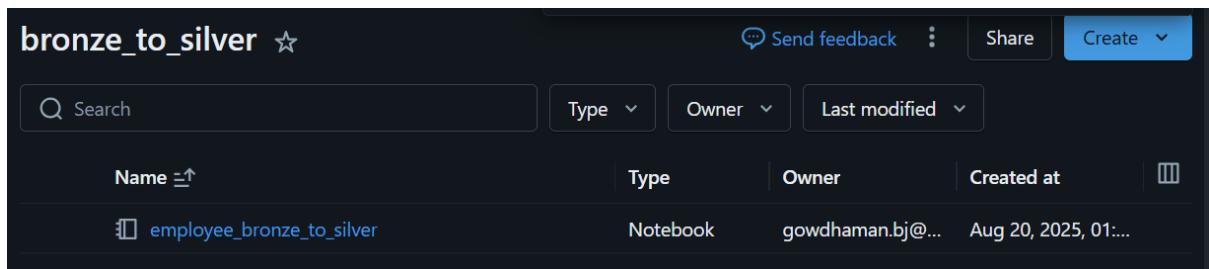
2. Create 4 notebooks in this respective order.

2 Notebooks named in source_to_bronze as utils (add all common functions in this notebook) and employee_source_to_bronze (driver notebook)



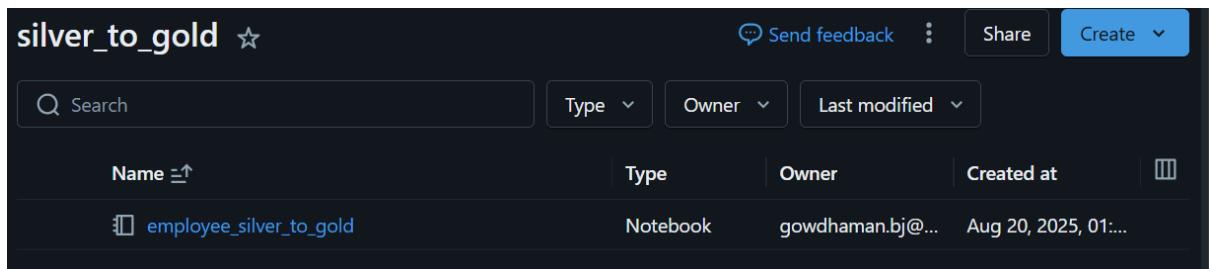
Name	Type	Owner	Created at
employee_source_to_bronze	Notebook	gowdhaman.bj@...	Aug 20, 2025, 01:...
utils	Notebook	gowdhaman.bj@...	Aug 20, 2025, 01:...

1 Notebook in bronze to silver as employee_bronze_to_silver



Name	Type	Owner	Created at
employee_bronze_to_silver	Notebook	gowdhaman.bj@...	Aug 20, 2025, 01:...

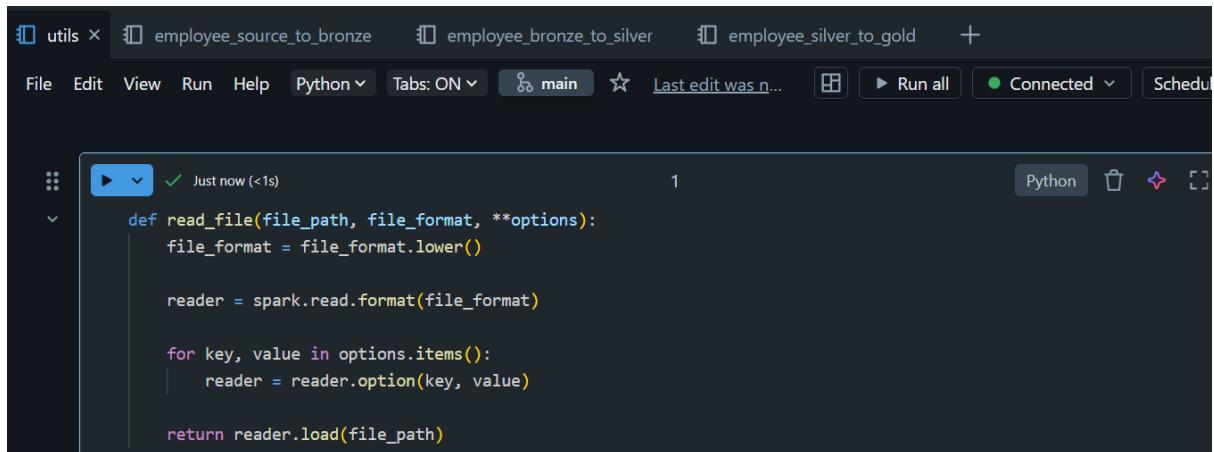
1 Notebook in silver to gold as employee_silver_to_gold



Name	Type	Owner	Created at
employee_silver_to_gold	Notebook	gowdhaman.bj@...	Aug 20, 2025, 01:...

3. Read the 3 datasets as Dataframe in employee_source_to_bronze, call utils notebook in this notebook, and write to a location in DBFS, as

/source_to_bronze/file_name.csv (employee, department_df, country_df) as CSV format.



utils x employee_source_to_bronze employee_bronze_to_silver employee_silver_to_gold +

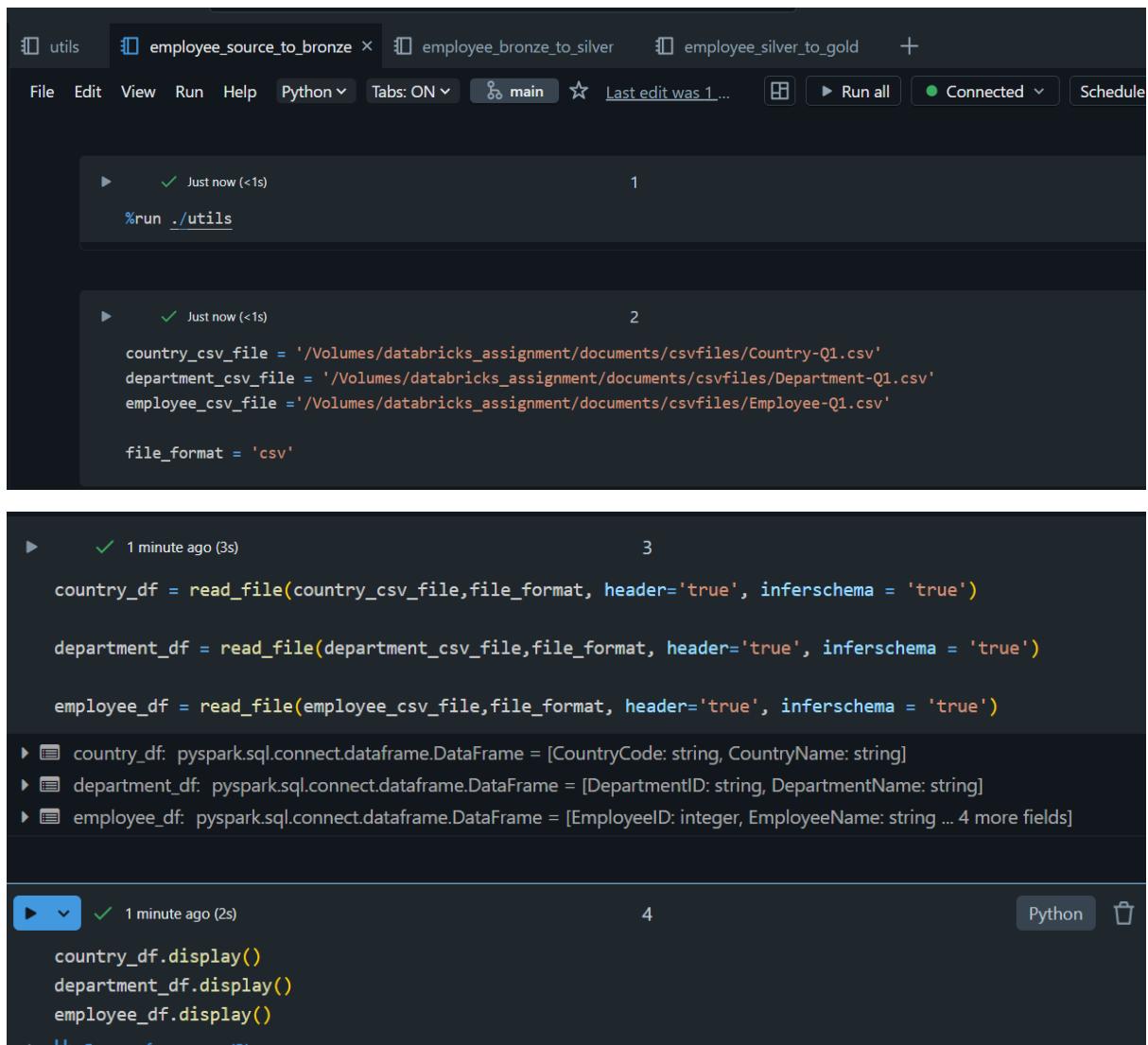
File Edit View Run Help Python Tabs: ON main Last edit was n... Run all Connected Schedule

```
▶ Just now (<1s) 1
def read_file(file_path, file_format, **options):
    file_format = file_format.lower()

    reader = spark.read.format(file_format)

    for key, value in options.items():
        reader = reader.option(key, value)

    return reader.load(file_path)
```



utils x employee_source_to_bronze x employee_bronze_to_silver employee_silver_to_gold +

File Edit View Run Help Python Tabs: ON main Last edit was 1... Run all Connected Schedule

```
▶ Just now (<1s) 1
%run ./utils
```

```
▶ Just now (<1s) 2
country_csv_file = '/Volumes/databricks_assignment/documents/csvfiles/Country-Q1.csv'
department_csv_file = '/Volumes/databricks_assignment/documents/csvfiles/Department-Q1.csv'
employee_csv_file = '/Volumes/databricks_assignment/documents/csvfiles/Employee-Q1.csv'

file_format = 'csv'
```

```
▶ 1 minute ago (3s) 3
country_df = read_file(country_csv_file, file_format, header='true', inferSchema = 'true')

department_df = read_file(department_csv_file, file_format, header='true', inferSchema = 'true')

employee_df = read_file(employee_csv_file, file_format, header='true', inferSchema = 'true')

▶ country_df: pyspark.sql.connect.DataFrame = [CountryCode: string, CountryName: string]
▶ department_df: pyspark.sql.connect.DataFrame = [DepartmentID: string, DepartmentName: string]
▶ employee_df: pyspark.sql.connect.DataFrame = [EmployeeID: integer, EmployeeName: string ... 4 more fields]
```

```
▶ 1 minute ago (2s) 4
country_df.display()
department_df.display()
employee_df.display()
↳ See performance (2)
```

	A _C DepartmentID	A _C DepartmentName
1	D101	Sales
2	D102	Marketing
3	D103	Finance
4	D104	Support
5	D105	HR

	A _C CountryCode	A _C CountryName
1	CN	China
2	IN	India
3	SA	South Africa
4	JA	Japan
5	MY	Malaysia
6	MA	Morocco

1 ² ₃ EmployeeID	A _C EmployeeName	A _C Department	A _C Country	1 ² ₃ Salary	1 ² ₃ Age
1	1 James	D101	IN	9000	25
2	2 Michel	D102	SA	8000	26
3	3 James son	D101	IN	10000	35
4	4 Robert	D103	MY	11000	34
5	5 Scott	D104	MA	6000	36
6	6 Gen	D105	JA	21345	24
7	7 John	D102	MY	87654	40
8	8 Maria	D105	SA	38144	38
9	9 Soffy	D103	IN	23456	29
10	10 Amy	D103	CN	21345	24

4. In **employee_bronze_to_silver**, call utils notebook in this notebook.
 Read the file located in DBFS location source_to_bronze with as data frame different read methods using custom schema.
 (dbfs access is not their)

```
%run /Repos/gowdhaman.bj@diggibyte.com/databricks_assignment/source_to_bronze/utils
```

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

▶ ✓ 3 minutes ago (<1s) 3

```
dept_schema = StructType([
    StructField("DepartmentID", IntegerType(), True),
    StructField("DepartmentName", StringType(), True)
])

employee_schema = StructType([
    StructField("EmployeeID", IntegerType(), True),
    StructField("EmployeeName", StringType(), True),
    StructField("Department", StringType(), True),
    StructField("Country", StringType(), True),
    StructField("Salary", IntegerType(), True),
    StructField("Age", IntegerType(), True)
])

country_schema = StructType([
    StructField("CountryCode", StringType(), True),
    StructField("CountryName", StringType(), True)
])
```

▶ ✓ 2 minutes ago (<1s) 4

```
country_csv_file = '/Volumes/databricks_assignment/documents/csvfiles/Country-Q1.csv'
department_csv_file = '/Volumes/databricks_assignment/documents/csvfiles/Department-Q1.csv'
employee_csv_file = '/Volumes/databricks_assignment/documents/csvfiles/Employee-Q1.csv'

file_format = 'csv'
```

▶ ✓ 1 minute ago (7s) 5

```
read_dept = read_file(department_csv_file,file_format,schema = dept_schema,header=True)
read_country = read_file(country_csv_file,file_format,schema=country_schema,header=True)
read_employee = read_file(employee_csv_file,file_format,schema=employee_schema,header=True)
display(read_dept)
display(read_country)
display(read_employee)
```

5. convert the Camel case of the columns to the snake case using UDF.

convert the Camel case of the columns to the snake case using UDF.

▶ ✓ 03:22 PM (<1s)

7

```
import re
```

▶ ✓ 5 minutes ago (<1s)

8

```
def camel_to_snake_case(camel_str):
    s1 = re.sub('([A-Z][a-z]+)', r'\1_\2', camel_str)
    return re.sub('([a-z0-9])([A-Z])', r'\1_\2', s1).lower()

def rename_columns_to_snake_case(df):
    new_cols = [camel_to_snake_case(col) for col in df.columns]
    return df.toDF(*new_cols)
```

▶ ✓ 03:38 PM (4s)

9

```
read_dept = rename_columns_to_snake_case(read_dept)
read_country = rename_columns_to_snake_case(read_country)
read_employee = rename_columns_to_snake_case(read_employee)

# Show results
display(read_dept)
display(read_country)
display(read_employee)
```

▶ See performance (3)

▶ read_dept: pyspark.sql.connect.dataframe.DataFrame = [department_id: string
▶ read_country: pyspark.sql.connect.dataframe.DataFrame = [country_code: strin
▶ read_employee: pyspark.sql.connect.dataframe.DataFrame = [employee_id: str

Table ▾ +

	A ^B C department_id	A ^B C department_name
1	D101	Sales
2	D102	Marketing

6. Add the **load_date** column with the current date.

The primary key is EmployeeID, the Database name is Employee_info, Table name is dim_employee.

write the DF as a delta table to the location /silver/db_name/table_name.

Add the load_date column with the current date.

A screenshot of a Databricks notebook cell. The code cell contains:

```
▶ ✓ Just now (2s) 11  
employee_load_date = read_employee.withColumn('load_date', current_date())  
display(employee_load_date)  
> See performance (1) Optimize  
▶ employee_load_date: pyspark.sql.connect.DataFrame = [employee_id: string, employee_name: string ... 5 more fields]
```

The resulting table view shows the following data:

employee_id	employee_name	department	country	salary	age	load_date
1	James	D101	IN	9000	25	2025-08-20
2	Michel	D102	SA	8000	26	2025-08-20
3	James son	D101	IN	10000	35	2025-08-20
4	Robert	D103	MY	11000	34	2025-08-20
5	Scott	D104	MA	6000	36	2025-08-20

The primary key is EmployeeID, the Database name is Employee_info, Table name is dim_employee.

write the DF as a delta table to the location /silver/db_name/table_name.

A screenshot of a Databricks notebook cell. The code cell contains:

```
▶ ✓ 3 minutes ago (8s) 13  
employee_load_date.write.format('delta').mode('overwrite').option("overwriteSchema",  
"true").saveAsTable('databricks_assignment.employee_info.dim_employee')  
> See performance (1) Optimize
```