
Sample CSV Data (employee_data.csv)

```
EmployeeID,Name,Department,JoiningDate,Salary
1001,John Doe,HR,2021-01-15,55000
1002,Jane Smith,IT,2020-03-10,62000
1003,Emily Johnson,Finance,2019-07-01,70000
1004,Michael Brown,HR,2018-12-22,54000
1005,David Wilson,IT,2021-06-25,58000
1006,Linda Davis,Finance,2020-11-15,67000
1007,James Miller,IT,2019-08-14,65000
1008,Barbara Moore,HR,2021-03-29,53000
```

Sample JSON Data (product_data.json)

```
[
  {
    "ProductID": 101,
    "ProductName": "Laptop",
    "Category": "Electronics",
    "Price": 1200,
    "Stock": 35
  },
  {
    "ProductID": 102,
    "ProductName": "Smartphone",
    "Category": "Electronics",
    "Price": 800,
    "Stock": 80
  },
  {
    "ProductID": 103,
    "ProductName": "Desk Chair",
    "Category": "Furniture",
    "Price": 150,
    "Stock": 60
  },
  {
    "ProductID": 104,
    "ProductName": "Monitor",
    "Category": "Electronics",
    "Price": 300,
    "Stock": 45
  },
  {
    "ProductID": 105,
    "ProductName": "Desk",
    "Category": "Furniture",
    "Price": 350,
    "Stock": 25
```

```
  }
]
```

Assignment 1: Working with CSV Data (employee_data.csv)

Tasks:

1. **Load the CSV data**:
   - Load the `employee_data.csv` file into a DataFrame.
   - Display the first 10 rows and inspect the schema.
2. **Data Cleaning**:
   - Remove rows where the `Salary` is less than 55,000.
   - Filter the employees who joined after the year 2020.
3. **Data Aggregation**:
   - Find the **average salary** by **Department**.
   - Count the number of employees in each **Department**.
4. **Write the Data to CSV**:
   - Save the cleaned data (from the previous steps) to a new CSV file.

Assignment 2: Working with JSON Data (product_data.json)

Tasks:

1. **Load the JSON data**:
   - Load the `product_data.json` file into a DataFrame.
   - Display the first 10 rows and inspect the schema.
2. **Data Cleaning**:
   - Remove rows where `Stock` is less than 30.
   - Filter the products that belong to the "Electronics" category.
3. **Data Aggregation**:
   - Calculate the **total stock** for products in the "Furniture" category.
   - Find the **average price** of all products in the dataset.
4. **Write the Data to JSON**:
   - Save the cleaned and aggregated data into a new JSON file.

Assignment 3: Working with Delta Tables

Tasks:

1. **Convert CSV and JSON Data to Delta Format**:
   - Convert the **employee_data.csv** and **product_data.json** into Delta Tables.
   - Save the Delta tables to a specified location.

2. **Register Delta Tables**:
    - Register both the employee and product Delta tables as SQL tables.
3. **Data Modifications with Delta Tables**:
    - Perform an **update** operation on the **employee** Delta table: Increase the salary by 5% for all employees in the **IT department**.
    - Perform a **delete** operation on the **product** Delta table: Delete products where the stock is less than 40.
4. **Time Travel with Delta Tables**:
    - Query the **product** Delta table to show its state before the delete operation (use **time travel**).
    - Retrieve the version of the **employee** Delta table before the salary update.
5. **Query Delta Tables**:
    - Query the **employee** Delta table to find the employees in the **Finance department**.
    - Query the **product** Delta table to find all products in the **Electronics** category with a price greater than 500.

Sample Code Snippets

1. Loading and Writing CSV Data

```
# Load CSV data
df_csv = spark.read.format("csv").option("header",
"true").load("/path_to_file/employee_data.csv")
df_csv.show()

# Filter and clean data
df_cleaned = df_csv.filter(df_csv['Salary'] > 55000)

# Write back to CSV
df_cleaned.write.format("csv").option("header",
"true").save("/path_to_output/cleaned_employee_data.csv")
```

2. Loading and Writing JSON Data

```
# Load JSON data
df_json = spark.read.format("json").load("/path_to_file/product_data.json")
df_json.show()

# Filter and clean data
df_filtered = df_json.filter(df_json['Stock'] > 30)

# Write back to JSON
df_filtered.write.format("json").save("/path_to_output/filtered_product_data.json")
```

3. Delta Table Operations

```
# Write DataFrame to Delta Table
df_csv.write.format("delta").mode("overwrite").save("/path_to_delta/employee_delta")

# Read Delta Table
df_delta = spark.read.format("delta").load("/path_to_delta/employee_delta")
df_delta.show()

# Update Delta Table
df_delta.createOrReplaceTempView("employee_delta")
spark.sql("""
    UPDATE employee_delta
    SET Salary = Salary * 1.05
    WHERE Department = 'IT'
""")

# Time Travel - Query a Previous Version
df_version = spark.read.format("delta").option("versionAsOf",
1).load("/path_to_delta/employee_delta")
df_version.show()
```

Deliverables

- CSV data with cleaned and filtered employee information.
- JSON data with filtered and aggregated product information.
- Delta tables for both employee and product data with updated, deleted, and time-traveled versions.
- SQL queries and results for Delta tables.
