Object of the Task

The primary objective of this task is to teach students how to:

1. **Map Data**: Understand how to align fields from different data formats (JSON, XML, CSV) into a unified structure.
2. **Transform Data**: Convert data from one format to another (e.g., XML to JSON, CSV to JSON).
3. **Combine Data**: Merge data from multiple sources into a single, consistent format.
4. **Save Data**: Store the unified data in a file for further use or analysis.

This task emphasizes the importance of **data interoperability** and **data integration**, which are critical skills in software development, data engineering, and system integration.

How This Task Relates to the Real World

1. **Mapping Fields**:
    - In the real world, different suppliers may use different field names (e.g., id vs. product_id, name vs. product_name).
    - The task teaches students how to map these fields into a consistent structure.
2. **Transforming Data**:
    - Suppliers may provide data in different formats (JSON, XML, CSV).
    - The task teaches students how to convert these formats into a unified format (e.g., JSON).
3. **Combining Data**:
    - In the real world, data from multiple sources must be combined into a single database or catalog.
    - The task teaches students how to merge data from different sources.
4. **Saving Data**:
    - The unified data must be saved in a file or database for further use.
    - The task teaches students how to save data in a file (e.g., unified_products.json).

Real-World Tools and Technologies

In the real world, this task is often accomplished using:

1. **Programming Languages**:
    - PHP, Python, Java, etc., for scripting and automation.
2. **Data Transformation Tools**:

    - ETL (Extract, Transform, Load) tools like **Talend**, **Informatica**, or **Apache NiFi**.
3. **Databases**:
    - Databases like MySQL, PostgreSQL, or MongoDB to store the unified data.
4. **APIs**:
    - APIs to fetch data from suppliers in real-time.

Key Takeaways for Students

1. **Data Mapping**:
    - Learn how to align fields from different formats into a unified structure.
2. **Data Transformation**:
    - Understand how to convert data from one format to another.
3. **Data Integration**:
    - Gain experience combining data from multiple sources.
4. **Real-World Applications**:
    - See how these skills are used in industries like e-commerce, healthcare, finance, and IoT.

Example of Real-World Impact

- **E-Commerce**: Unified product catalogs improve customer experience and streamline operations.
- **Healthcare**: Unified patient records enable better diagnosis and treatment.
- **Finance**: Consolidated transaction data helps in fraud detection and financial analysis.
- **IoT**: Unified sensor data enables real-time monitoring and decision-making.

By completing this task, students will gain hands-on experience with **data mapping**, **transformation**, and **integration**, which are essential skills for real-world software development and data engineering.

Let me know if you need further clarification or additional examples!

Below are **real-world examples** and **practical use cases** for **Data Mapping and Exchange** that you can demonstrate to your students. These examples will help them understand the importance of data mapping and exchange in various industries and applications.

1. E-Commerce Product Data Integration

Scenario

An e-commerce platform needs to integrate product data from multiple suppliers. Each supplier provides product information in different formats (JSON, XML, CSV). The platform needs to map and transform this data into a unified format for its database.

Example

- **Supplier A (JSON)**:

```
json
```

```
{
  "products": [
    {"id": 101, "name": "Smartphone", "price": 599.99, "category": "Electronics"},
    {"id": 102, "name": "Laptop", "price": 999.99, "category": "Electronics"}
  ]
}
```

- **Supplier B (XML)**:

```
xml
```

```
<products>
  <product>
    <id>201</id>
    <name>Headphones</name>
    <price>49.99</price>
    <category>Accessories</category>
  </product>
  <product>
    <id>202</id>
    <name>Smartwatch</name>
    <price>199.99</price>
    <category>Wearables</category>
  </product>
</products>
```

- **Supplier C (CSV)**:

```
Copy
```

```
id,name,price,category
301,Tablet,299.99,Electronics
302,Printer,149.99,Office
```

Task for Students

- Write scripts to map and transform the data from all three suppliers into a unified JSON format.
- Save the unified data into a single file (unified_products.json).

Task Solution

Step 1: Prepare the Input Files

1. **Supplier A (JSON)** - Save as supplier_a.json:

```
json
```

```
{
  "products": [
    {"id": 101, "name": "Smartphone", "price": 599.99, "category": "Electronics"},
    {"id": 102, "name": "Laptop", "price": 999.99, "category": "Electronics"}
  ]
}
```

2. **Supplier B (XML)** - Save as supplier_b.xml:

```
xml
```

```
<products>
  <product>
    <id>201</id>
    <name>Headphones</name>
    <price>49.99</price>
    <category>Accessories</category>
  </product>
  <product>
    <id>202</id>
    <name>Smartwatch</name>
    <price>199.99</price>
    <category>Wearables</category>
  </product>
</products>
```

3. **Supplier C (CSV)** - Save as supplier_c.csv:

```
id,name,price,category
301,Tablet,299.99,Electronics
302,Printer,149.99,Office
```

Step 2: Write the PHP Script

Here's the PHP script to map and transform the data from all three suppliers into a unified JSON format:

```php
<?php
// Function to load JSON data
function loadJsonData($file) {
    $json = file_get_contents($file);
    return json_decode($json, true);
}

// Function to load XML data
function loadXmlData($file) {
    $xml = simplexml_load_file($file);
    $products = [];
    foreach ($xml->product as $product) {
        $products[] = [
            'id' => (int)$product->id,
            'name' => (string)$product->name,
            'price' => (float)$product->price,
            'category' => (string)$product->category
        ];
    }
    return $products;
}

// Function to load CSV data
function loadCsvData($file) {
    $products = [];
    if (($handle = fopen($file, 'r')) !== false) {
        $header = fgetcsv($handle); // Read header
        while (($row = fgetcsv($handle)) !== false) {
            $products[] = [
                'id' => (int)$row[0],
                'name' => $row[1],
                'price' => (float)$row[2],
                'category' => $row[3]
```

```
            ];
        }
        fclose($handle);
    }
    return $products;
}

// Load data from all suppliers
$supplierA = loadJsonData('supplier_a.json')['products'];
$supplierB = loadXmlData('supplier_b.xml');
$supplierC = loadCsvData('supplier_c.csv');

// Combine all data into a single array
$unifiedProducts = array_merge($supplierA, $supplierB, $supplierC);

// Save unified data to a JSON file
file_put_contents('unified_products.json', json_encode(['products' => $unifiedProducts], JSON_
PRETTY_PRINT));

echo "Data mapping and transformation complete. Check unified_products.json.\n";
?>
```

Scenario

A weather forecasting system collects weather data from multiple sources (JSON, XML). The system needs to map and consolidate this data for analysis.

Example

- **Source A (JSON)**:

  ```
  json

  Copy
  ```

  ```
  {
    "weather_data": [
      {"location": "New York", "temperature": 20.5, "humidity": 60},
      {"location": "Los Angeles", "temperature": 25.0, "humidity": 50}
    ]
  }
  ```

- **Source B (XML)**:

  ```
  xml
  ```

Copy

```
<weather>
  <record>
    <location>Chicago</location>
    <temperature>18.0</temperature>
    <humidity>70</humidity>
  </record>
  <record>
    <location>Miami</location>
    <temperature>28.0</temperature>
    <humidity>75</humidity>
  </record>
</weather>
```