Sure thing! Let's tackle this step by step.

### Step 1: Create Tables and Insert Rows

First, let's create two tables and insert five rows into each:

```sql

-- Create Table1

CREATE TABLE Table1 (

ID INT PRIMARY KEY,

Name NVARCHAR(50),

Age INT

);

-- Insert rows into Table1

INSERT INTO Table1 (ID, Name, Age)

VALUES

(1, 'John Doe', 30),

(2, 'Jane Smith', 25),

(3, 'Tom Brown', 40),

(4, 'Linda White', 35),

(5, 'Steve Black', 28);

-- Create Table2

CREATE TABLE Table2 (

ID INT PRIMARY KEY,

Address NVARCHAR(100),

City NVARCHAR(50)

);

-- Insert rows into Table2

INSERT INTO Table2 (ID, Address, City)

VALUES

(1, '123 Elm St', 'Springfield'),

(2, '456 Oak St', 'Rivertown'),

(3, '789 Pine St', 'Lakeside'),

(4, '101 Maple St', 'Mountainview'),

(5, '202 Birch St', 'Riverdale');

```

### Step 2: Join Tables and Get Data as XML and JSON

Next, let's join the two tables and get the data as XML and JSON:

```sql

-- Join Tables and Get Data as XML

SELECT

Table1.ID,

Table1.Name,

Table1.Age,

Table2.Address,

Table2.City

FROM Table1

JOIN Table2 ON Table1.ID = Table2.ID

FOR XML PATH('Row'), ROOT('Data');

-- Join Tables and Get Data as JSON

SELECT

Table1.ID,

Table1.Name,

Table1.Age,

Table2.Address,

Table2.City

FROM Table1

JOIN Table2 ON Table1.ID = Table2.ID

FOR JSON AUTO, ROOT('Data');

```

### Step 3: Parse XML/JSON and Create Virtual Table

Let's now create a template for parsing XML/JSON as `NVARCHAR(MAX)` and inserting it into a virtual table:

```sql

-- Example XML/JSON Data as NVARCHAR(MAX)

DECLARE @xmlData NVARCHAR(MAX) =

'<Data>

<Row>

<ID>1</ID>

<Name>John Doe</Name>

<Age>30</Age>

<Address>123 Elm St</Address>

<City>Springfield</City>

</Row>

...

</Data>';

DECLARE @jsonData NVARCHAR(MAX) =

'{

"Data": [

{

"ID": 1,

"Name": "John Doe",

"Age": 30,

"Address": "123 Elm St",

"City": "Springfield"

},

...

]

}';

-- Parse XML Data

WITH XMLTable AS (

SELECT

XmlData.value('ID[1]', 'INT') AS ID,

XmlData.value('Name[1]', 'NVARCHAR(50)') AS Name,

XmlData.value('Age[1]', 'INT') AS Age,

XmlData.value('Address[1]', 'NVARCHAR(100)') AS Address,

XmlData.value('City[1]', 'NVARCHAR(50)') AS City

FROM

(SELECT CAST(@xmlData AS XML) AS XmlContent) AS Data

CROSS APPLY XmlContent.nodes('/Data/Row') AS XmlData(XmlData)

)

SELECT \* FROM XMLTable;

-- Parse JSON Data

SELECT \*

FROM OPENJSON(@jsonData, '$.Data')

WITH (

ID INT,

Name NVARCHAR(50),

Age INT,

Address NVARCHAR(100),

City NVARCHAR(50)

);

```

### Step 4: Create Exercises

#### Exercise 1: Add New Node to JSON/XML and Parse

Assuming the manufacturer added a new node in the JSON/XML, here is the updated structure:

\*\*Updated XML:\*\*

```xml

<Data>

<Row>

<ID>1</ID>

<Name>John Doe</Name>

<Age>30</Age>

<Address>123 Elm St</Address>

<City>Springfield</City>

<Country>USA</Country>

</Row>

...

</Data>

```

\*\*Updated JSON:\*\*

```json

{

"Data": [

{

"ID": 1,

"Name": "John Doe",

"Age": 30,

"Address": "123 Elm St",

"City": "Springfield",

"Country": "USA"

},

...

]

}

```

Now, parse the updated XML/JSON and put the data into a virtual table:

```sql

-- Parse Updated XML Data

WITH UpdatedXMLTable AS (

SELECT

XmlData.value('ID[1]', 'INT') AS ID,

XmlData.value('Name[1]', 'NVARCHAR(50)') AS Name,

XmlData.value('Age[1]', 'INT') AS Age,

XmlData.value('Address[1]', 'NVARCHAR(100)') AS Address,

XmlData.value('City[1]', 'NVARCHAR(50)') AS City,

XmlData.value('Country[1]', 'NVARCHAR(50)') AS Country

FROM

(SELECT CAST(@xmlData AS XML) AS XmlContent) AS Data

CROSS APPLY XmlContent.nodes('/Data/Row') AS XmlData(XmlData)

)

SELECT \* FROM UpdatedXMLTable;

-- Parse Updated JSON Data

DECLARE @updatedJsonData NVARCHAR(MAX) =

'{

"Data": [

{

"ID": 1,

"Name": "John Doe",

"Age": 30,

"Address": "123 Elm St",

"City": "Springfield",

"Country": "USA"

},

...

]

}';

SELECT \*

FROM OPENJSON(@updatedJsonData, '$.Data')

WITH (

ID INT,

Name NVARCHAR(50),

Age INT,

Address NVARCHAR(100),

City NVARCHAR(50),

Country NVARCHAR(50)

);

```

#### Exercise 2: Export Data in Specified XML/JSON Format

Here are two example structures for XML and JSON. Try exporting the data in these formats:

\*\*XML Structure:\*\*

```xml

<Company>

<Employee>

<EmployeeID>1</EmployeeID>

<FullName>John Doe</FullName>

<Details>

<Age>30</Age>

<Address>123 Elm St</Address>

<City>Springfield</City>

<Country>USA</Country>

</Details>

</Employee>

...

</Company>

```

\*\*JSON Structure:\*\*

```json

{

"Company": {

"Employees": [

{

"EmployeeID": 1,

"FullName": "John Doe",

"Details": {

"Age": 30,

"Address": "123 Elm St",

"City": "Springfield",

"Country": "USA"

}

},

...

]

}

}

```

Now, create SQL queries to export data in these formats.

Let me know if you need any further help or clarification!