ASSIGNMENT-4

1.Convert bookstore.xml into json

XML:

```
<bookstore>
  <book>
    <title>Harry Potter</title>
    <author>J.K. Rowling</author>
    <price>29.99</price>
    <available>true</available>
  </book>
  <book>
    <title>The Hobbit</title>
    <author>J.R.R. Tolkien</author>
    <price>19.99</price>
    <available>false</available>
  </book>
</bookstore>
```

JSON:

```
{
  "bookstore": {
    "book": [
        {
          "title": "Harry Potter",
          "author": "J.K. Rowling",
          "price": 29.99,
          "available": true
        },
        {
          "title": "The Hobbit",
          "author": "J.R.R. Tolkien",
          "price": 19.99,
          "available": false
        }
      ]
    }
}
```

2)Write a query to give inner join,left outer join,right outer join and full outer join

Tables:

Employee Table:

employee_id	first_name	last_name	department_id
1	John	Doe	10
2	Jane	Smith	20

3	Mike	Johnson	30
4	Emily	Davis	10

Department Table:

department_id	department_name
10	HR
20	Sales
30	IT
40	Marketing

1. INNER JOIN

An inner join returns only the rows where there is a match in both tables.

Query:

```sql
SELECT e.employee_id, e.first_name, e.last_name, d.department_name
FROM Employee e
INNER JOIN Department d ON e.department_id = d.department_id;
```

Result:

employee_id	first_name	last_name	department_name
1	John	Doe	HR
2	Jane	Smith	Sales

3	Mike	Johnson	IT
4	Emily	Davis	HR

2. LEFT OUTER JOIN

A left outer join returns all rows from the left table (Employee), and the matched rows from the right table (Department). If no match is found, NULL values are returned for columns from the right table.

Query:

```sql
SELECT e.employee_id, e.first_name, e.last_name, d.department_name
FROM Employee e
LEFT OUTER JOIN Department d ON e.department_id = d.department_id;
```

Result:

employee_id	first_name	last_name	department_name
1	John	Doe	HR
2	Jane	Smith	Sales
3	Mike	Johnson	IT
4	Emily	Davis	HR

3. RIGHT OUTER JOIN

A right outer join returns all rows from the right table (Department), and the matched rows from the left table (Employee). If no match is found, NULL values are returned for columns from the left table.

Query:

```sql
SELECT e.employee_id, e.first_name, e.last_name, d.department_name
FROM Employee e
RIGHT OUTER JOIN Department d ON e.department_id = d.department_id;
```

Result:

employee_id	first_name	last_name	department_name
1	John	Doe	HR
2	Jane	Smith	Sales
3	Mike	Johnson	IT
4	Emily	Davis	HR
NULL	NULL	NULL	Marketing

4. FULL OUTER JOIN

A full outer join returns all rows when there is a match in either left (Employee) or right (Department) table records. If there is no match, the result is NULL from the side where there is no match.

Query:

```sql
SELECT e.employee_id, e.first_name, e.last_name, d.department_name
FROM Employee e
FULL OUTER JOIN Department d ON e.department_id = d.department_id;
```

Result:

employee_id	first_name	last_name	department_name
1	John	Doe	HR
2	Jane	Smith	Sales
3	Mike	Johnson	IT
4	Emily	Davis	HR
NULL	NULL	NULL	Marketing

3)Write a query to find duplicate records

1. Based on `first_name`

Query:

```sql
SELECT first_name, COUNT(*)
FROM Employee
GROUP BY first_name
HAVING COUNT(*) > 1;
```

2. Based on `email`

Query:

```sql
SELECT email, COUNT(*)
FROM Employee
GROUP BY email
HAVING COUNT(*) > 1;
```

3. Based on `first_name` and `last_name`

Query:

```sql
SELECT first_name, last_name, COUNT(*)
FROM Employee
GROUP BY first_name, last_name
HAVING COUNT(*) > 1;
```

4. Based on `first_name` and `email`

Query:

```sql
SELECT first_name, email, COUNT(*)
FROM Employee
GROUP BY first_name, email
HAVING COUNT(*) > 1;
```

Given the sample data:

employee_id	first_name	last_name	email
1	John	Doe	john.doe@example.com
2	Jane	Smith	jane.smith@example.com
3	John	Doe	john.doe@example.com
4	Emily	Davis	emily.davis@example.com

Results

1. Based on `first_name`:

Result:

```
| first_name | COUNT(*) |
|------------|----------|
| John       | 2        |
```

2. Based on email:

Result:

```
| email                 | COUNT(*) |
|-----------------------|----------|
| john.doe@example.com  | 2        |
```

3. Based on `first_name` and `last_name`:

Result:

```
| first_name | last_name | COUNT(*) |
|------------|-----------|----------|
| John       | Doe       | 2        |
```

4. Based on `first_name` and `email`:

Result:

```
| first_name | email | COUNT(*) |
|------------|-----------------------|----------|
| John       | john.doe@example.com | 2  |
```