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 |

Employee Table with email:

| 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|

1. INNER JOIN:

An inner join in a database matches up data from two tables based on a shared attribute, only keeping the rows where there's a match in both tables. It's like finding where two sets of information overlap and combining them.

SELECT Employee.employee\_id, Employee.first\_name, Employee.last\_name, Employee.department\_id, Department.department\_name

FROM Employee

INNER JOIN Department ON Employee.department\_id = Department.department\_id;

Result:

| employee\_id | first\_name | last\_name | department\_id | department\_name |

|-------------|------------|-----------|---------------|-----------------|

| 1 | John | Doe | 10 | HR |

| 2 | Jane | Smith | 20 | Sales |

| 3 | Mike | Johnson | 30 | IT |

| 4 | Emily | Davis | 10 | HR |

1. LEFT OUTER JOIN:

A left outer join in a database pairs up data from two tables based on a common attribute, including all rows from the left table and only matching rows from the right table. It ensures that all data from the left table is retained, even if there are no matches in the right table.

SELECT Employee.employee\_id, Employee.first\_name, Employee.last\_name, Employee.department\_id, Department.department\_name

FROM Employee

LEFT OUTER JOIN Department ON Employee.department\_id = Department.department\_id;

Result:

| employee\_id | first\_name | last\_name | department\_id | department\_name |

|-------------|------------|-----------|---------------|-----------------|

| 1 | John | Doe | 10 | HR |

| 2 | Jane | Smith | 20 | Sales |

| 3 | Mike | Johnson | 30 | IT |

| 4 | Emily | Davis | 10 | HR |

1. RIGHT OUTER JOIN:

A right outer join in a database pairs up data from two tables based on a shared attribute, including all rows from the right table and only matching rows from the left table. It ensures that all data from the right table is retained, even if there are no matches in the left table.

SELECT Employee.employee\_id, Employee.first\_name, Employee.last\_name, Employee.department\_id, Department.department\_name

FROM Employee

RIGHT OUTER JOIN Department ON Employee.department\_id = Department.department\_id;

Result:

| employee\_id | first\_name | last\_name | department\_id | department\_name |

|-------------|------------|-----------|---------------|-----------------|

| 1 | John | Doe | 10 | HR |

| 2 | Jane | Smith | 20 | Sales |

| 3 | Mike | Johnson | 30 | IT |

| NULL | NULL | NULL | 40 | Marketing |

1. FULL OUTER JOIN:

A full outer join in a database combines data from two tables based on a shared attribute, including all rows from both tables. It ensures that all data from both tables is retained, matching rows where possible and filling in with nulls where there are no matches.

SELECT Employee.employee\_id, Employee.first\_name, Employee.last\_name, Employee.department\_id, Department.department\_name

FROM Employee

FULL OUTER JOIN Department ON Employee.department\_id = Department.department\_id;

Result:

| employee\_id | first\_name | last\_name | department\_id | department\_name |

|-------------|------------|-----------|---------------|-----------------|

| 1 | John | Doe | 10 | HR |

| 2 | Jane | Smith | 20 | Sales |

| 3 | Mike | Johnson | 30 | IT |

| 4 | Emily | Davis | 10 | HR |

| NULL | NULL | NULL | 40 | Marketing |

#eliminating duplicate files based on different attributes

1) Based on firstName:

SELECT first\_name, COUNT(\*)

FROM Employee

GROUP BY first\_name

HAVING COUNT(\*) > 1;

Result:

| first\_name | COUNT(\*) |

|------------|----------|

| John | 2 |

2) Based on email:

SELECT email, COUNT(\*)

FROM Employee

GROUP BY email

HAVING COUNT(\*) > 1;

Result:

| email | COUNT(\*) |

|----------------------|----------|

| john.doe@example.com | 2 |

3) Based on firstname and Last Name:

SELECT first\_name, last\_name, COUNT(\*)

FROM Employee

GROUP BY first\_name, last\_name

HAVING COUNT(\*) > 1;

Result:

| first\_name | last\_name | COUNT(\*) |

|------------|-----------|----------|

| John | Doe | 2 |

4) Based on firstname and email:

SELECT first\_name, email, COUNT(\*)

FROM Employee

GROUP BY first\_name, email

HAVING COUNT(\*) > 1;

Result:

| first\_name | email | COUNT(\*) |

|------------|----------------------|----------|

| John | john.doe@example.com | 2

|

1)Convert bookstore.xml into json

XML File

<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 File

{

"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

}

]

}

}