**Relational Algebra Operations in SQL (Oracle) - Answer Key with Notations**

**Scenario:** You are managing a university database with two tables:

- STUDENT(SID, Name, DeptID)
- DEPARTMENT(DeptID, DeptName)

Some students may not be assigned to a department yet, and some departments may have no students.

---

**1. Table Creation and Sample Data (Run this in Oracle Live SQL)**

```
-- Drop tables if they already exist

DROP TABLE STUDENT;

DROP TABLE DEPARTMENT;


-- Create Department Table

CREATE TABLE DEPARTMENT (

  DeptID NUMBER PRIMARY KEY,

  DeptName VARCHAR2(50)

);


-- Create Student Table

CREATE TABLE STUDENT (

  SID NUMBER PRIMARY KEY,

  Name VARCHAR2(50),

  DeptID NUMBER REFERENCES DEPARTMENT(DeptID)

);


-- Insert sample data

INSERT INTO DEPARTMENT VALUES (10, 'Computer Science');

INSERT INTO DEPARTMENT VALUES (20, 'Mathematics');

INSERT INTO DEPARTMENT VALUES (30, 'Physics');

INSERT INTO DEPARTMENT VALUES (40, 'Chemistry');
```

INSERT INTO STUDENT VALUES (1, 'Alice', 10);

INSERT INTO STUDENT VALUES (2, 'Bob', 20);

INSERT INTO STUDENT VALUES (3, 'Charlie', NULL);

INSERT INTO STUDENT VALUES (4, 'David', 10);

INSERT INTO STUDENT VALUES (5, 'Eva', 30);

---

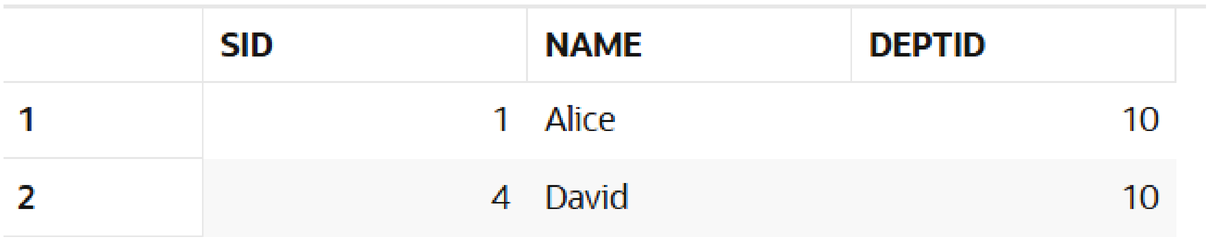
**2. Relational Algebra Operations - SQL Answers with Notations**

A. **Selection** (Select all students from the Computer Science department)

- **Relational Algebra**:

Relational Algebra: $\sigma_{DeptID=10}(STUDENT)$

SELECT * FROM STUDENT WHERE DeptID = 10;



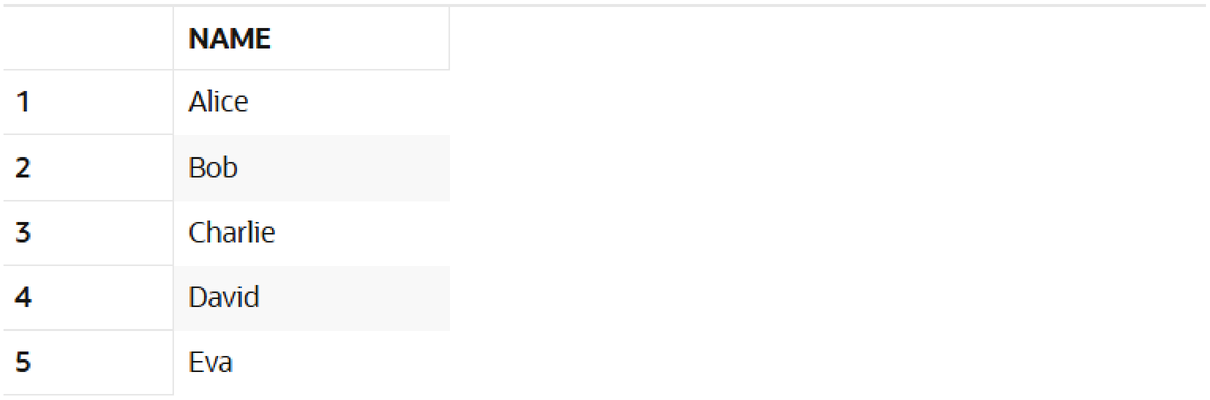
| | SID | NAME | DEPTID |
|---|---|---|---|
| 1 | 1 | Alice | 10 |
| 2 | 4 | David | 10 |

B. **Projection** (List only the names of students)

Relational Algebra: $\pi_{Name}(STUDENT)$

- **Relational Algebra**:

SELECT Name FROM STUDENT;



| | NAME |
|---|---|
| 1 | Alice |
| 2 | Bob |
| 3 | Charlie |
| 4 | David |
| 5 | Eva |

C. **Cartesian Product** (Show all combinations of students and departments)

Relational Algebra: $STUDENT \times DEPARTMENT$

- **Relational Algebra**:

SELECT * FROM STUDENT, DEPARTMENT;

**WARNING AS IT IS TOO BIG**

D. **Left Outer Join** (Show all students and their departments, even those without departments)

Relational Algebra: $STUDENT ⟕ DEPARTMENT$

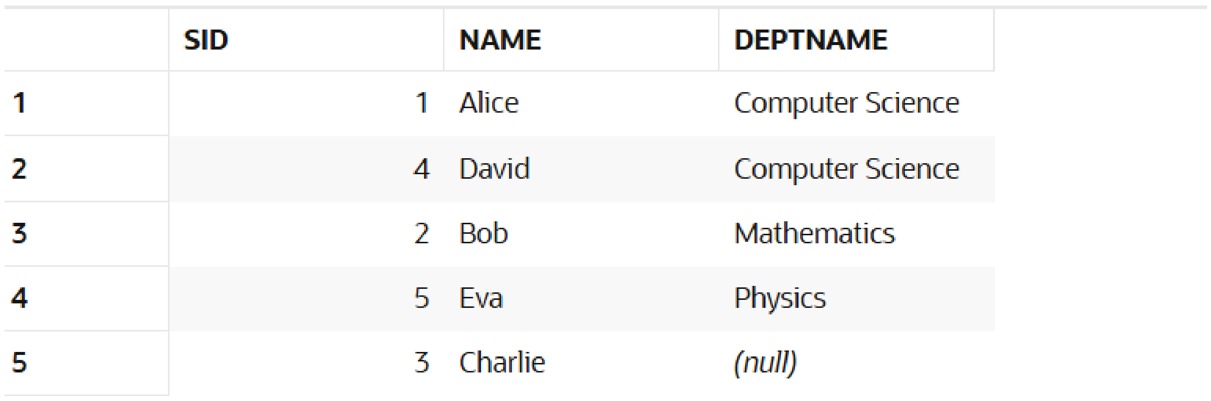
- **Relational Algebra**:

SELECT S.SID, S.Name, D.DeptName

FROM STUDENT S

LEFT OUTER JOIN DEPARTMENT D

ON S.DeptID = D.DeptID;



| | SID | NAME | DEPTNAME |
|---|---|---|---|
| 1 | 1 | Alice | Computer Science |
| 2 | 4 | David | Computer Science |
| 3 | 2 | Bob | Mathematics |
| 4 | 5 | Eva | Physics |
| 5 | 3 | Charlie | *(null)* |

E. **Right Outer Join** (Show all departments and their students, even those without students)

Relational Algebra: $STUDENT ⟖ DEPARTMENT$

- **Relational Algebra**:

SELECT S.SID, S.Name, D.DeptName

FROM STUDENT S

RIGHT OUTER JOIN DEPARTMENT D

ON S.DeptID = D.DeptID;

| | SID | NAME | DEPTNAME |
|---|---|---|---|
| **1** | 1 | Alice | Computer Science |
| **2** | 2 | Bob | Mathematics |
| **3** | 4 | David | Computer Science |
| **4** | 5 | Eva | Physics |
| **5** | *(null)* | *(null)* | Chemistry |

F. **Full Outer Join** (Show all students and departments with possible matches)

Relational Algebra: $STUDENT ⟗ DEPARTMENT$

- **Relational Algebra**:

SELECT S.SID, S.Name, D.DeptName

FROM STUDENT S

FULL OUTER JOIN DEPARTMENT D

ON S.DeptID = D.DeptID;

| | SID | NAME | DEPTNAME |
|---|---|---|---|
| **1** | 1 | Alice | Computer Science |
| **2** | 2 | Bob | Mathematics |
| **3** | 3 | Charlie | *(null)* |
| **4** | 4 | David | Computer Science |
| **5** | 5 | Eva | Physics |
| **6** | *(null)* | *(null)* | Chemistry |

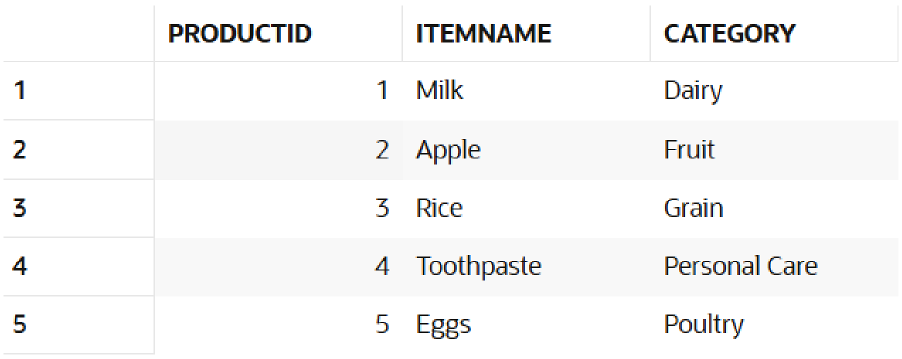
```
-- Create Product Table

DROP TABLE PRODUCT;

DROP TABLE PRODUCT_DETAIL;


CREATE TABLE PRODUCT (

   ProductID NUMBER PRIMARY KEY,

   ItemName VARCHAR2(100),

   Category VARCHAR2(50)

);


-- Create Product Detail Table

CREATE TABLE PRODUCT_DETAIL (

   ProductID NUMBER PRIMARY KEY,

   Brand VARCHAR2(100),

   Weight VARCHAR2(20),

   ExpiryDate DATE,

   StockQuantity NUMBER,

   Price NUMBER(6,2),

   FOREIGN KEY (ProductID) REFERENCES PRODUCT(ProductID)

);


INSERT INTO PRODUCT VALUES (1, 'Milk', 'Dairy');

INSERT INTO PRODUCT VALUES (2, 'Apple', 'Fruit');

INSERT INTO PRODUCT VALUES (3, 'Rice', 'Grain');

INSERT INTO PRODUCT VALUES (4, 'Toothpaste', 'Personal Care');

INSERT INTO PRODUCT VALUES (5, 'Eggs', 'Poultry');


INSERT INTO PRODUCT_DETAIL VALUES (1, 'Dutch Lady', '1L', TO_DATE('2025-08-15', 'YYYY-MM-DD'),
100, 4.50);

INSERT INTO PRODUCT_DETAIL VALUES (2, 'Envy', '200g', TO_DATE('2025-06-30', 'YYYY-MM-DD'), 50,
1.20);
```

INSERT INTO PRODUCT_DETAIL VALUES (3, 'Jasmine', '5kg', TO_DATE('2026-01-01', 'YYYY-MM-DD'), 75, 28.90);

INSERT INTO PRODUCT_DETAIL VALUES (4, 'Colgate', '150g', TO_DATE('2026-05-10', 'YYYY-MM-DD'), 200, 6.80);

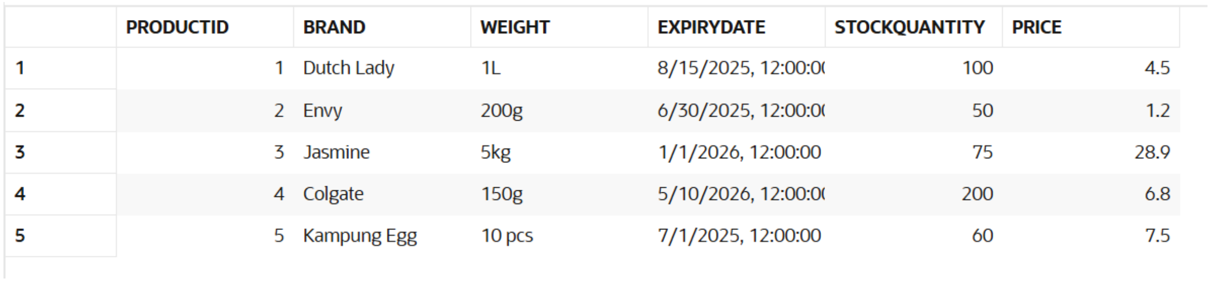
INSERT INTO PRODUCT_DETAIL VALUES (5, 'Kampung Egg', '10 pcs', TO_DATE('2025-07-01', 'YYYY-MM-DD'), 60, 7.50);

SELECT * FROM Product;



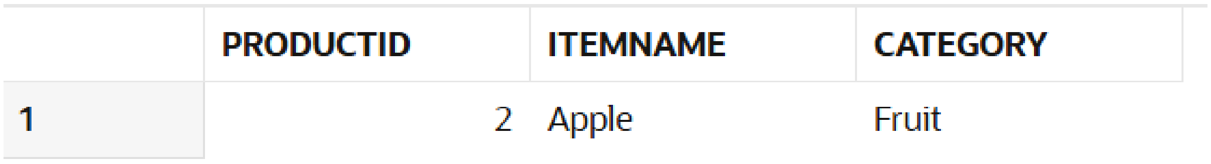
| | PRODUCTID | ITEMNAME | CATEGORY |
|---|---|---|---|
| 1 | 1 | Milk | Dairy |
| 2 | 2 | Apple | Fruit |
| 3 | 3 | Rice | Grain |
| 4 | 4 | Toothpaste | Personal Care |
| 5 | 5 | Eggs | Poultry |

SELECT * FROM Product_Detail;



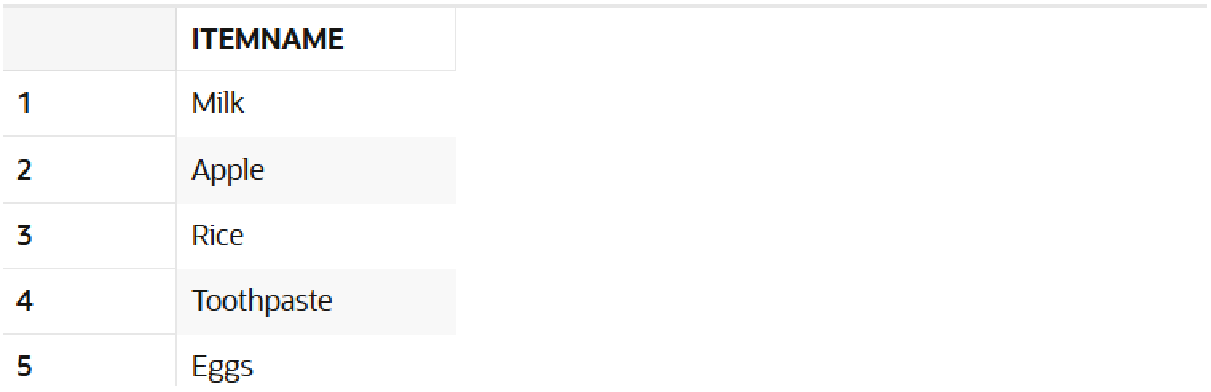
| | PRODUCTID | BRAND | WEIGHT | EXPIRYDATE | STOCKQUANTITY | PRICE |
|---|---|---|---|---|---|---|
| 1 | 1 | Dutch Lady | 1L | 8/15/2025, 12:00:0 | 100 | 4.5 |
| 2 | 2 | Envy | 200g | 6/30/2025, 12:00:0 | 50 | 1.2 |
| 3 | 3 | Jasmine | 5kg | 1/1/2026, 12:00:00 | 75 | 28.9 |
| 4 | 4 | Colgate | 150g | 5/10/2026, 12:00:0 | 200 | 6.8 |
| 5 | 5 | Kampung Egg | 10 pcs | 7/1/2025, 12:00:00 | 60 | 7.5 |

SELECT * FROM PRODUCT WHERE Category = 'Fruit'; ***SELECTION***



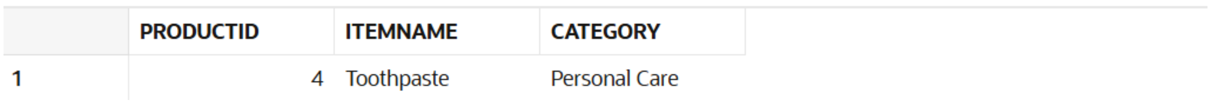
| | PRODUCTID | ITEMNAME | CATEGORY |
|---|---|---|---|
| 1 | 2 | Apple | Fruit |

SELECT ITEMNAME FROM PRODUCT ***PROJECTION***



| | ITEMNAME |
|---|---|
| 1 | Milk |
| 2 | Apple |
| 3 | Rice |
| 4 | Toothpaste |
| 5 | Eggs |

SELECT * FROM PRODUCT WHERE ProductID = 4; ***SELECTION***



| | PRODUCTID | ITEMNAME | CATEGORY |
|---|---|---|---|
| 1 | 4 | Toothpaste | Personal Care |

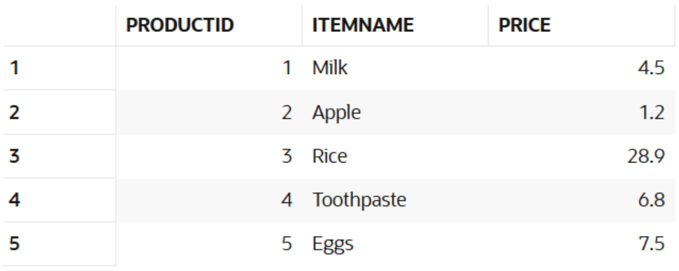
***LEFT OUTER JOIN***

SELECT P.ProductID, P.ItemName, D.Price

FROM Product P

LEFT OUTER JOIN Product_Detail D

ON P.ProductID = D.ProductID;



| | PRODUCTID | ITEMNAME | PRICE |
|---|---|---|---|
| 1 | 1 | Milk | 4.5 |
| 2 | 2 | Apple | 1.2 |
| 3 | 3 | Rice | 28.9 |
| 4 | 4 | Toothpaste | 6.8 |
| 5 | 5 | Eggs | 7.5 |

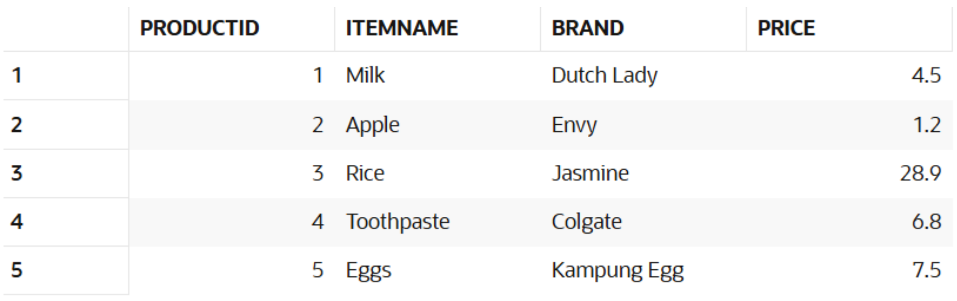
***RIGHT OUTER JOIN***

SELECT P.ProductID, P.ItemName, D.Brand, D.Price

FROM Product P

RIGHT OUTER JOIN Product_Detail D

ON P.ProductID = D.ProductID;



| | PRODUCTID | ITEMNAME | BRAND | PRICE |
|---|---|---|---|---|
| 1 | 1 | Milk | Dutch Lady | 4.5 |
| 2 | 2 | Apple | Envy | 1.2 |
| 3 | 3 | Rice | Jasmine | 28.9 |
| 4 | 4 | Toothpaste | Colgate | 6.8 |
| 5 | 5 | Eggs | Kampung Egg | 7.5 |

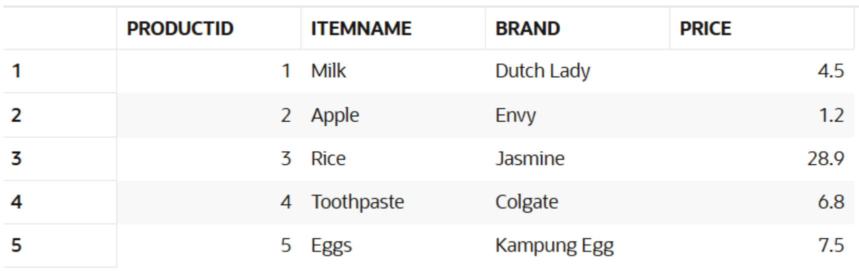
***FULL OUTER JOIN***

SELECT P.ProductID, P.ItemName, D.Brand, D.Price

FROM Product P

FULL OUTER JOIN Product_Detail D

ON P.ProductID = D.ProductID;



| | PRODUCTID | ITEMNAME | BRAND | PRICE |
|---|---|---|---|---|
| 1 | 1 | Milk | Dutch Lady | 4.5 |
| 2 | 2 | Apple | Envy | 1.2 |
| 3 | 3 | Rice | Jasmine | 28.9 |
| 4 | 4 | Toothpaste | Colgate | 6.8 |
| 5 | 5 | Eggs | Kampung Egg | 7.5 |

In this case, both outer join will be same as everything is completed and no null value

ALTER TABLE PRODUCT RENAME COLUMN ItemName TO ProductName;

DESCRIBE Product;

```
Name        Null?    Type
----------- -------- -------------
PRODUCTID   NOT NULL NUMBER
PRODUCTNAME          VARCHAR2(100)
CATEGORY             VARCHAR2(50)
```