DataBase Tutorial

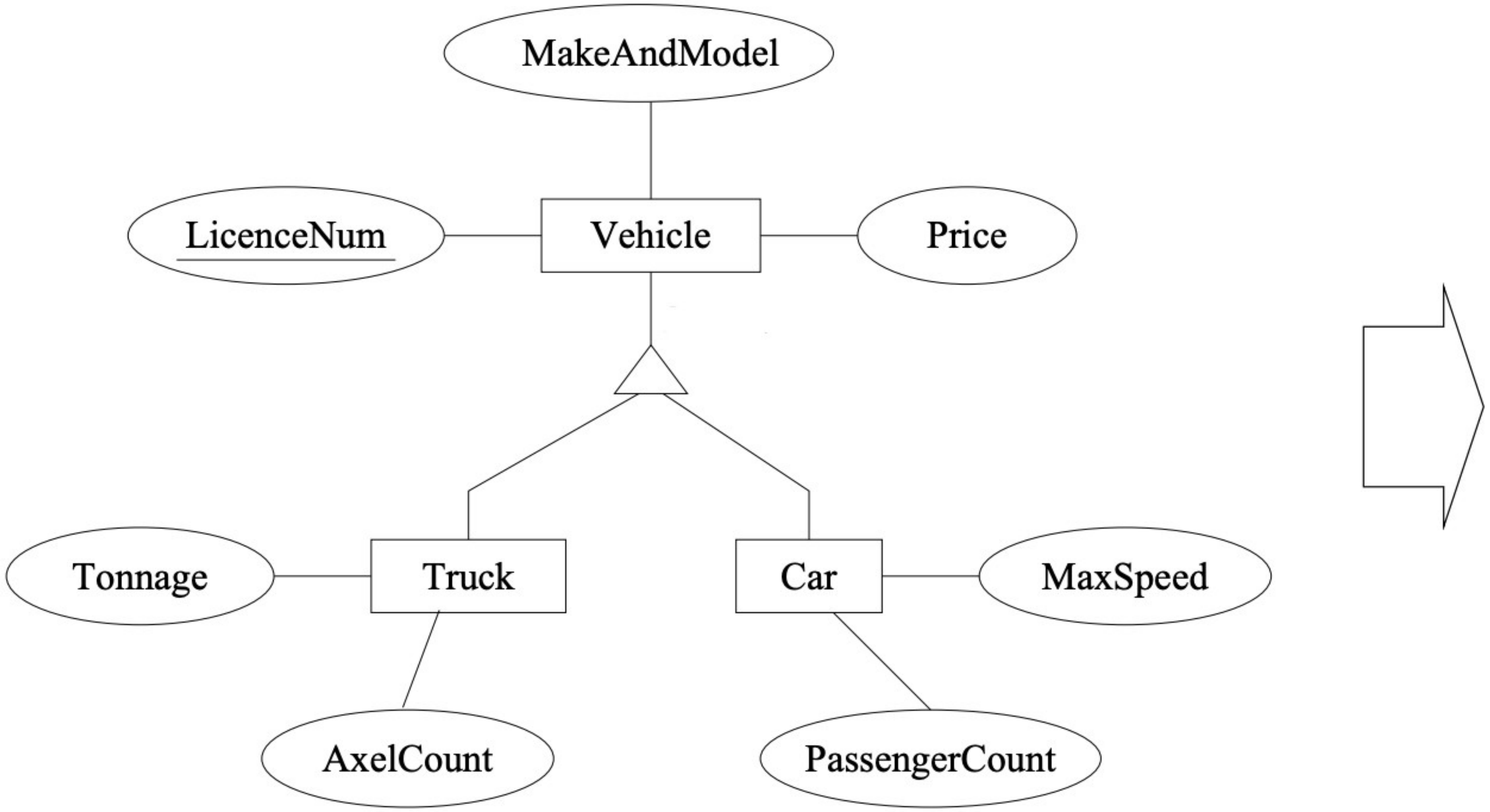
Sep. 27, 2021

Morteza Alipour

Outline

- ER Schema Mapping
- Aggregation Function
- Group by/Having/Joins/Views
- Questions

IS-A



Vehicle

LicenceNum	MakeAndModel	Price

Truck

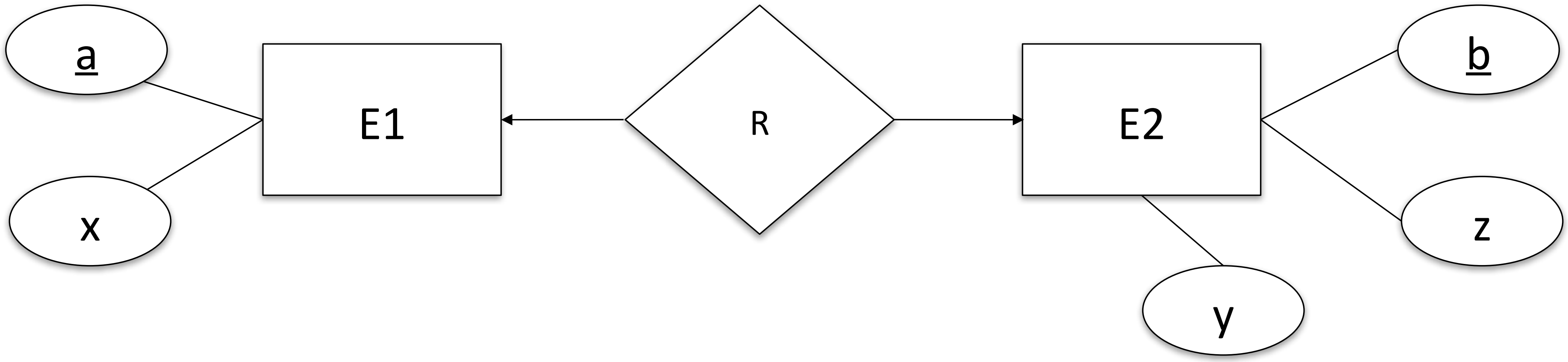
LicenceNum	Tonnage	AxelCount

Car

LicenceNum	MaxSpeed	PassengerCount

Source: https://cs.uwaterloo.ca/~gweddell/cs348/errelational-present.pdf

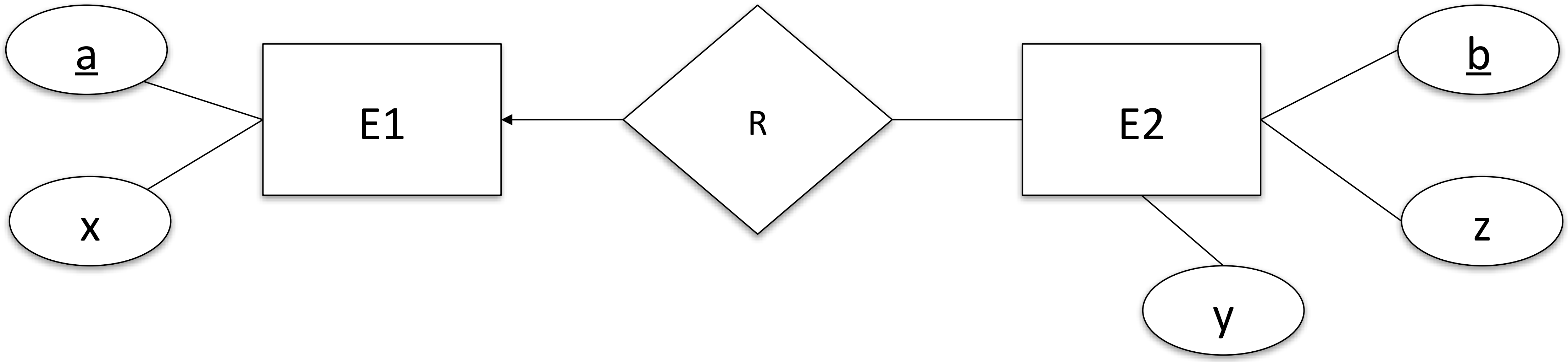
One-to-One Relationship



E1: (a, x, b), E2: (b, y, z)

Or E1: (a, x), E2: (b, y, z, a)

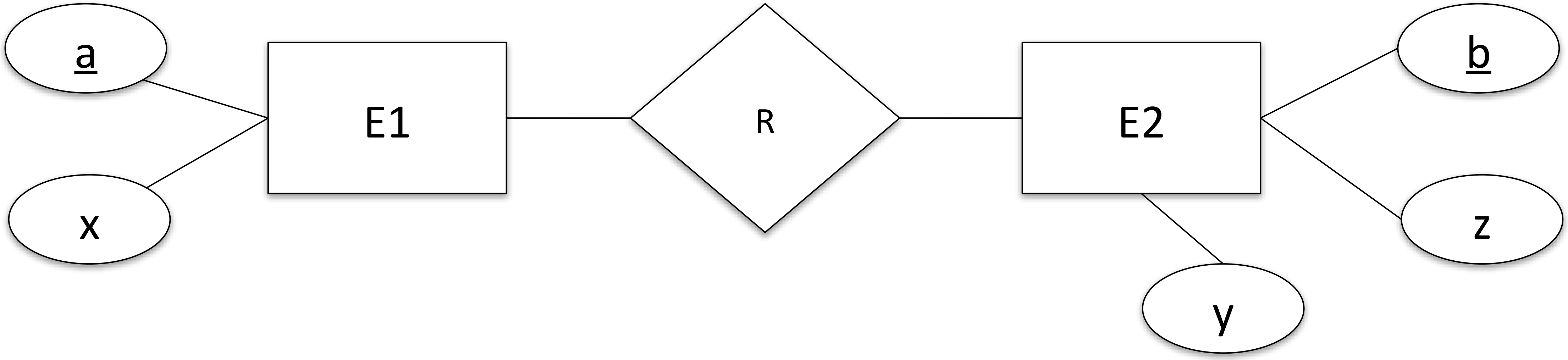
Many-to-One Relationship



E1: (a, x), E2: (b, y, z, a)

Note: the primary key of E1 is the foreign key of E2

Many-to-Many Relationship



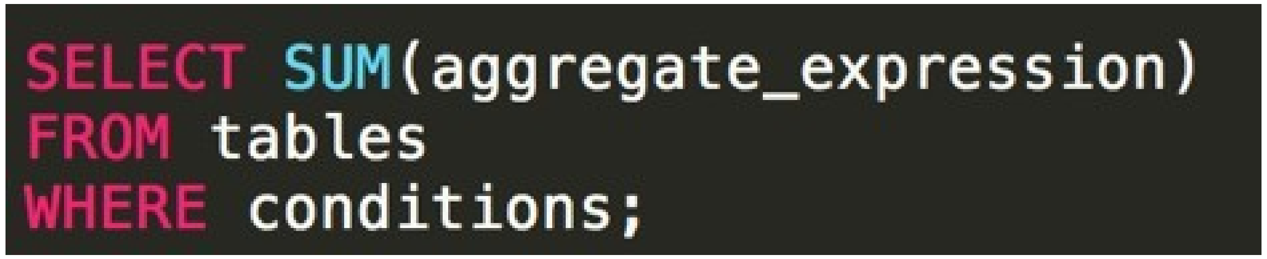
E1: (a, x), E2: (b, y, z), R: (a, b)

Aggregation Functions

- Aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single output value.
- Max, Min, Sum, Count, Avg
- Aggregate functions often need an added GROUP BY statement.

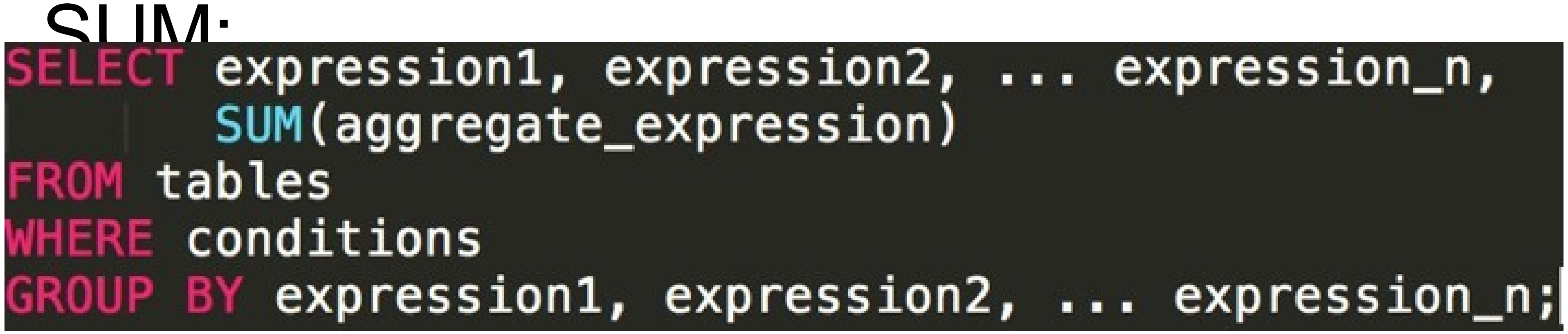
Example

- If you only want to return the SUM:



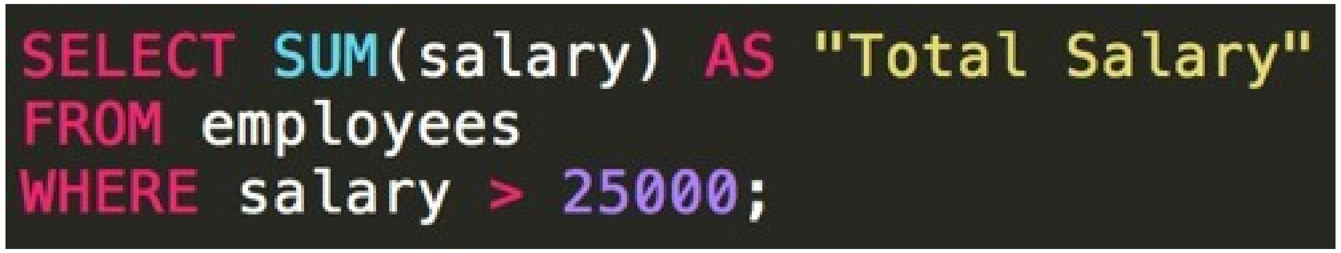
```
SELECT SUM(aggregate_expression)
FROM tables
WHERE conditions;
```

- If you want to return the several attributes and SUM:



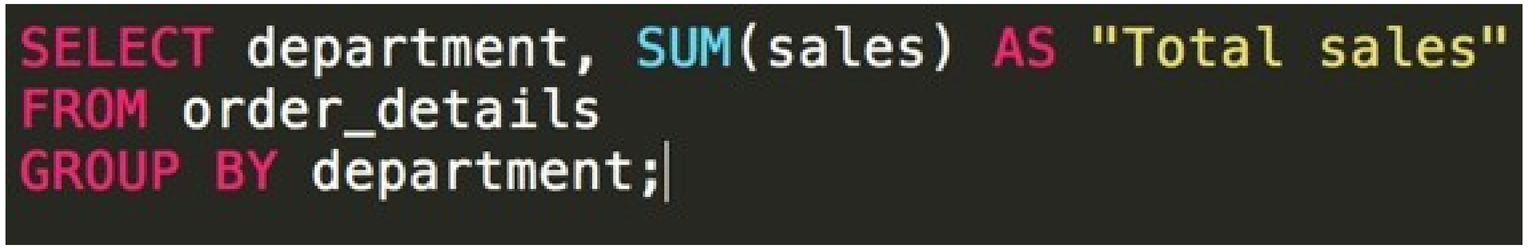
```
SELECT expression1, expression2, ... expression_n,
       SUM(aggregate_expression)
FROM tables
WHERE conditions
GROUP BY expression1, expression2, ... expression_n;
```

- Find out salary of all employees whose salary is above $25,000 / year. (Only return SUM)



```
SELECT SUM(salary) AS "Total Salary"
FROM employees
WHERE salary > 25000;
```

- Return the name of the department and the total sales (in the associated department).

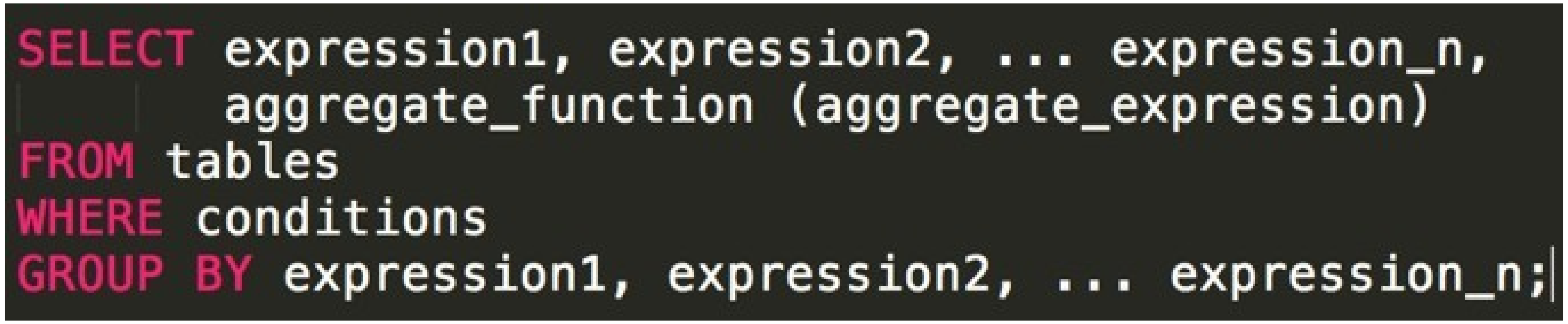


```
SELECT department, SUM(sales) AS "Total sales"
FROM order_details
GROUP BY department;
```

Group by

- The SQL GROUP BY clause can be used in a SELECT statement to collect data across multiple records and group the results by one or more columns.

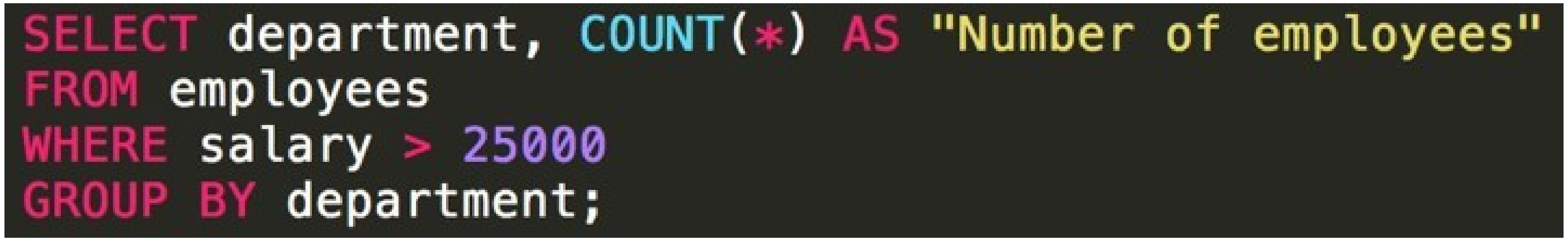
- It is often used with Aggregation Functions



```
SELECT expression1, expression2, ... expression_n,
       aggregate_function (aggregate_expression)
FROM tables
WHERE conditions
GROUP BY expression1, expression2, ... expression_n;
```

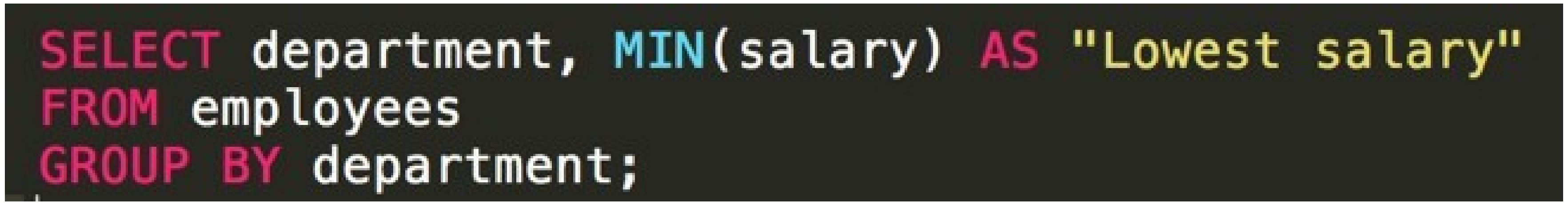
Example: group by and Aggregation Functions

- Uses the COUNT function to return the department and the number of employees (in the department) that make over $25,000 / year.



```
SELECT department, COUNT(*) AS "Number of employees"
FROM employees
WHERE salary > 25000
GROUP BY department;
```

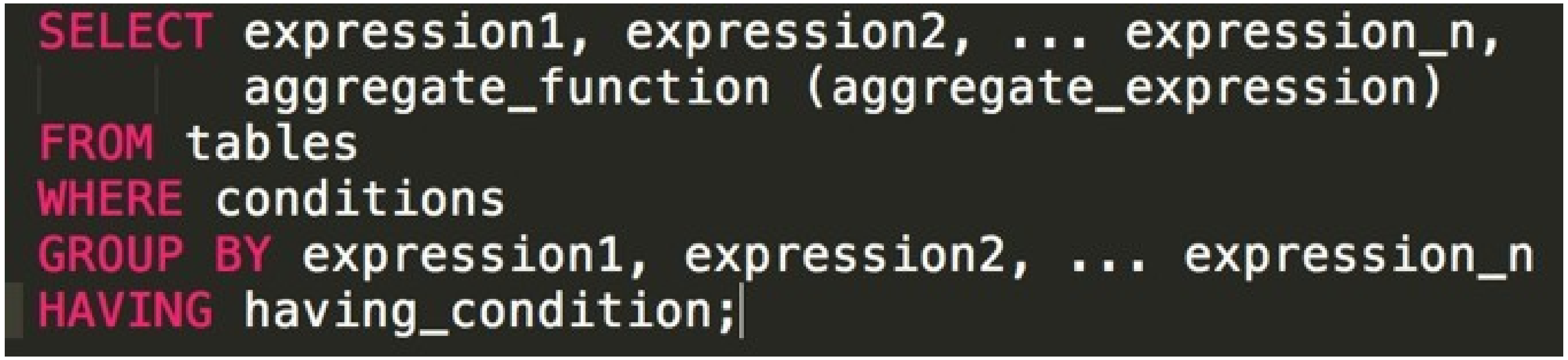
- uses the MIN function to return the name of each department and the minimum salary in the department.



```
SELECT department, MIN(salary) AS "Lowest salary"
FROM employees
GROUP BY department;
```

Having

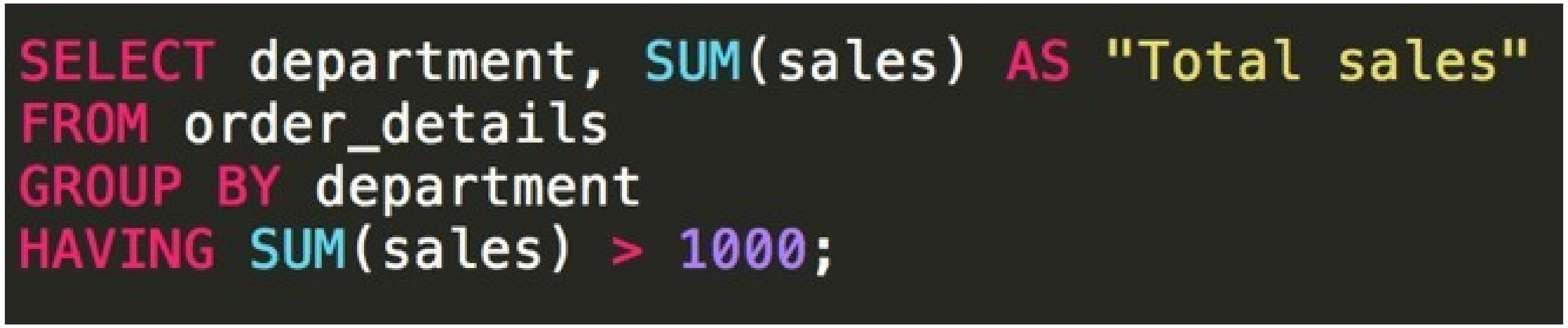
- The SQL HAVING Clause is used in combination with the GROUP BY Clause to restrict the groups of returned rows to only those whose the condition is TRUE.



```
SELECT expression1, expression2, ... expression_n,
       aggregate_function (aggregate_expression)
FROM tables
WHERE conditions
GROUP BY expression1, expression2, ... expression_n
HAVING having_condition;
```

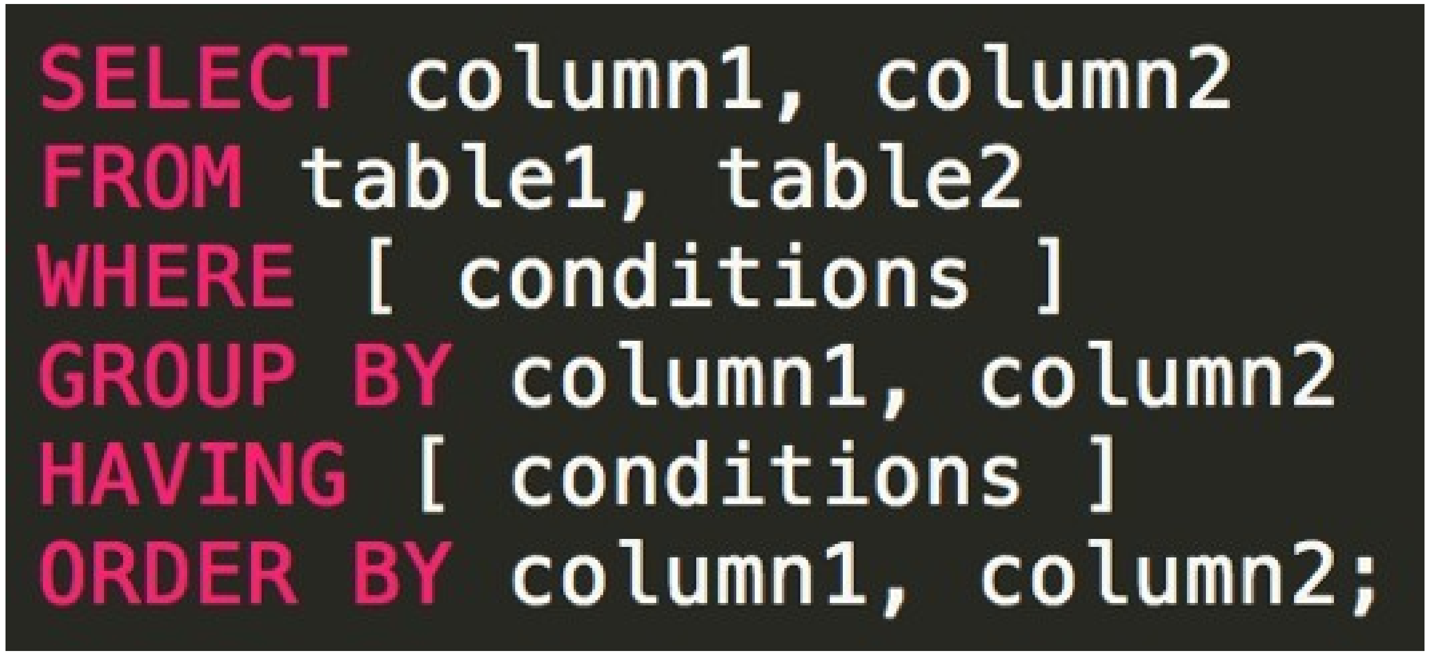
Example

- use the SQL SUM function to return the name of the department and the total sales (in the associated department). The SQL HAVING clause will filter the results so that only departments with sales greater than $1000 will be returned.



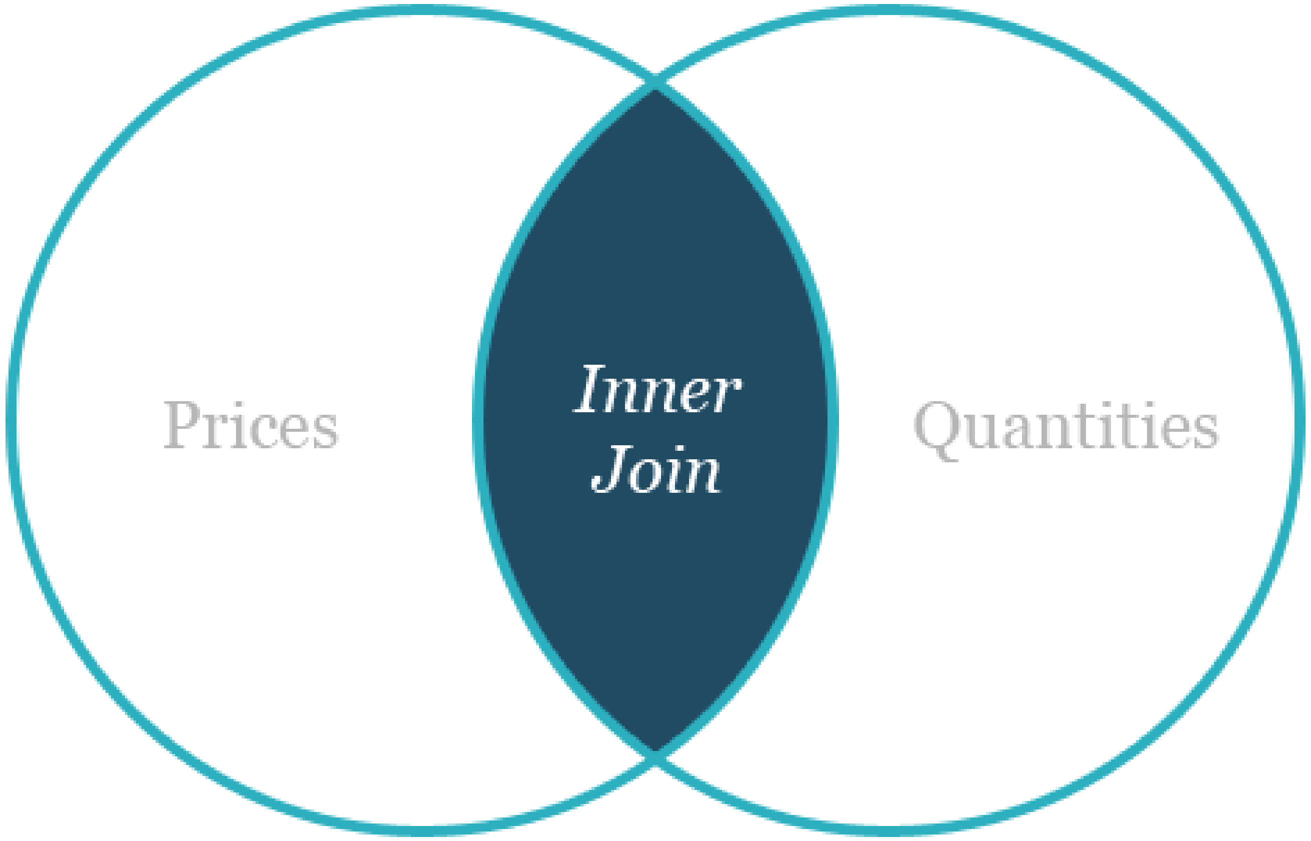
```sql
SELECT department, SUM(sales) AS "Total sales"
FROM order_details
GROUP BY department
HAVING SUM(sales) > 1000;
```

Statement Order



```
SELECT column1, column2
FROM table1, table2
WHERE [ conditions ]
GROUP BY column1, column2
HAVING [ conditions ]
ORDER BY column1, column2;
```

Inner Join



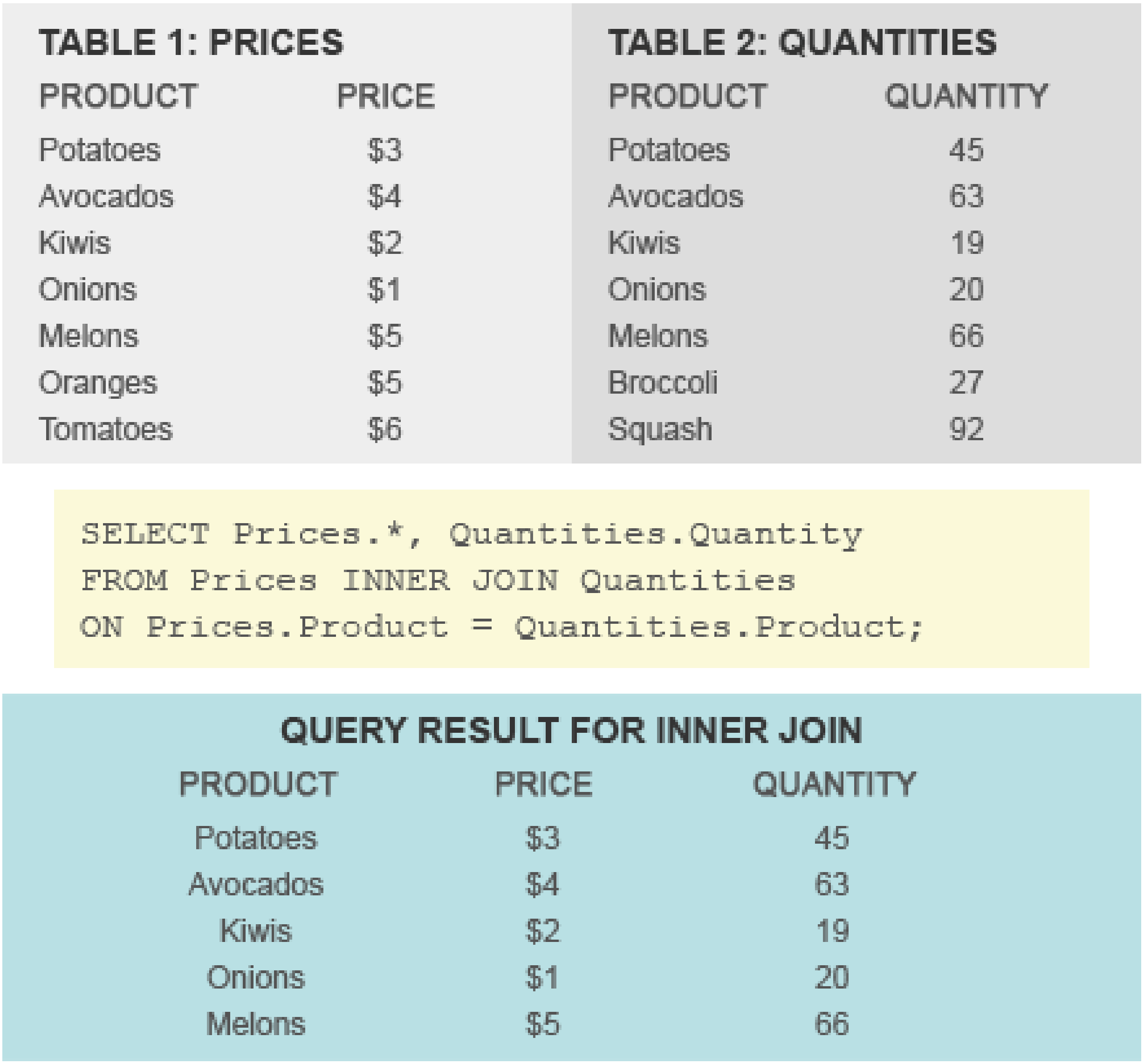


TABLE 1: PRICES

PRODUCT	PRICE
Potatoes	$3
Avocados	$4
Kiwis	$2
Onions	$1
Melons	$5
Oranges	$5
Tomatoes	$6

TABLE 2: QUANTITIES

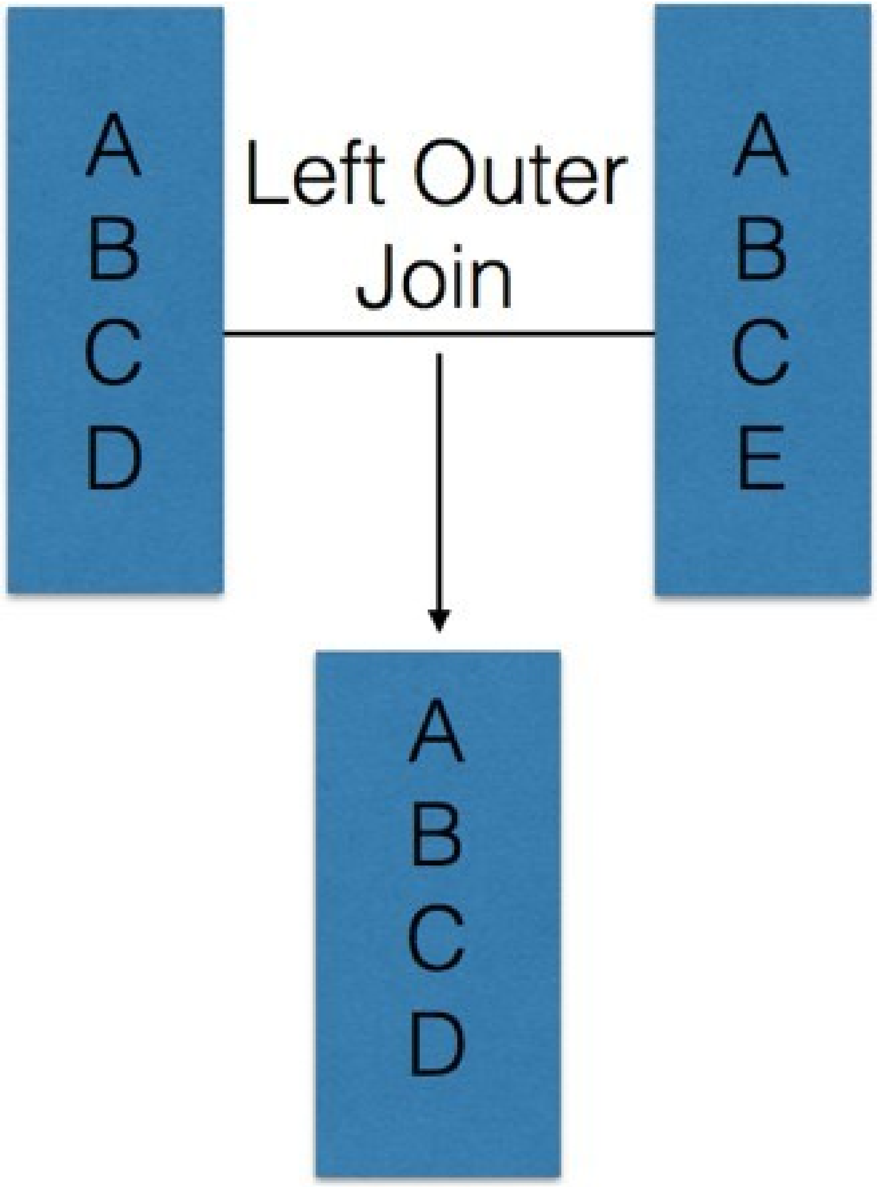
PRODUCT	QUANTITY
Potatoes	45
Avocados	63
Kiwis	19
Onions	20
Melons	66
Broccoli	27
Squash	92

```
SELECT Prices.*, Quantities.Quantity
FROM Prices INNER JOIN Quantities
ON Prices.Product = Quantities.Product;
```

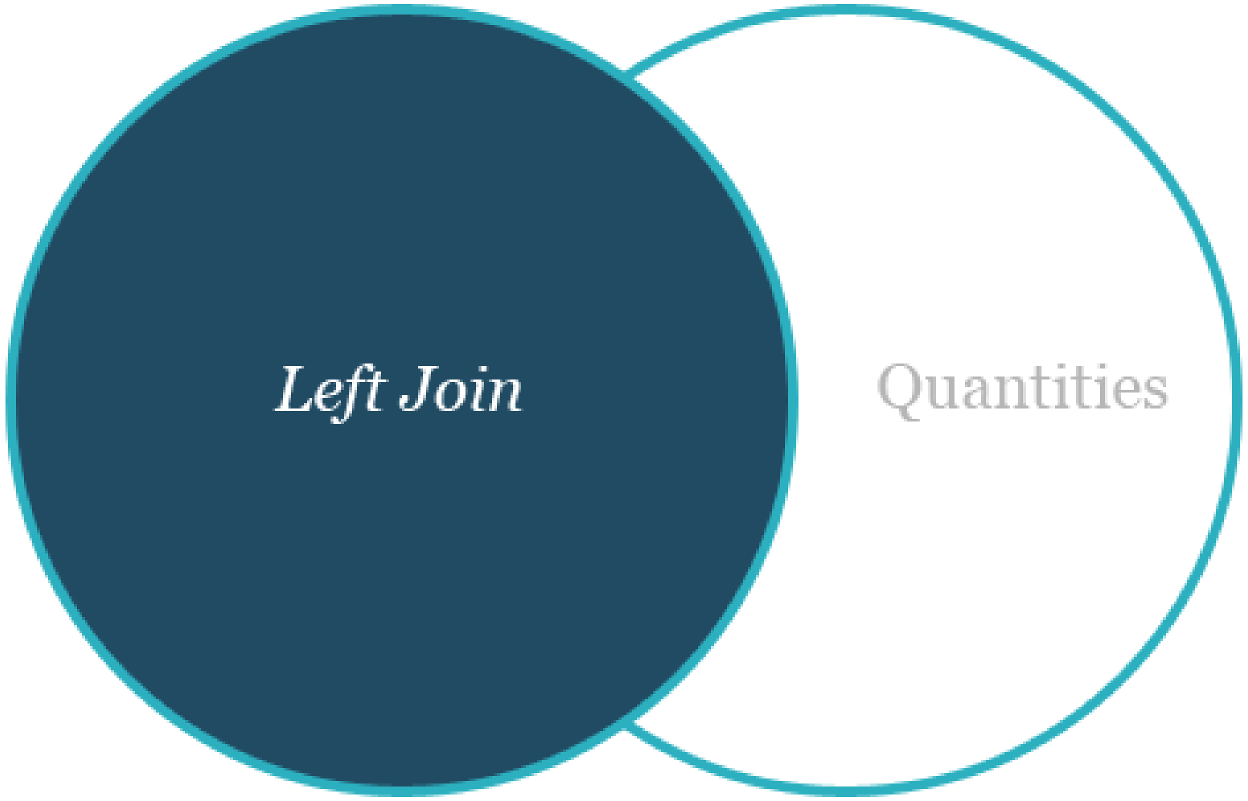
QUERY RESULT FOR INNER JOIN

PRODUCT	PRICE	QUANTITY
Potatoes	$3	45
Avocados	$4	63
Kiwis	$2	19
Onions	$1	20
Melons	$5	66

Left Outer Joins



Left Outer Join



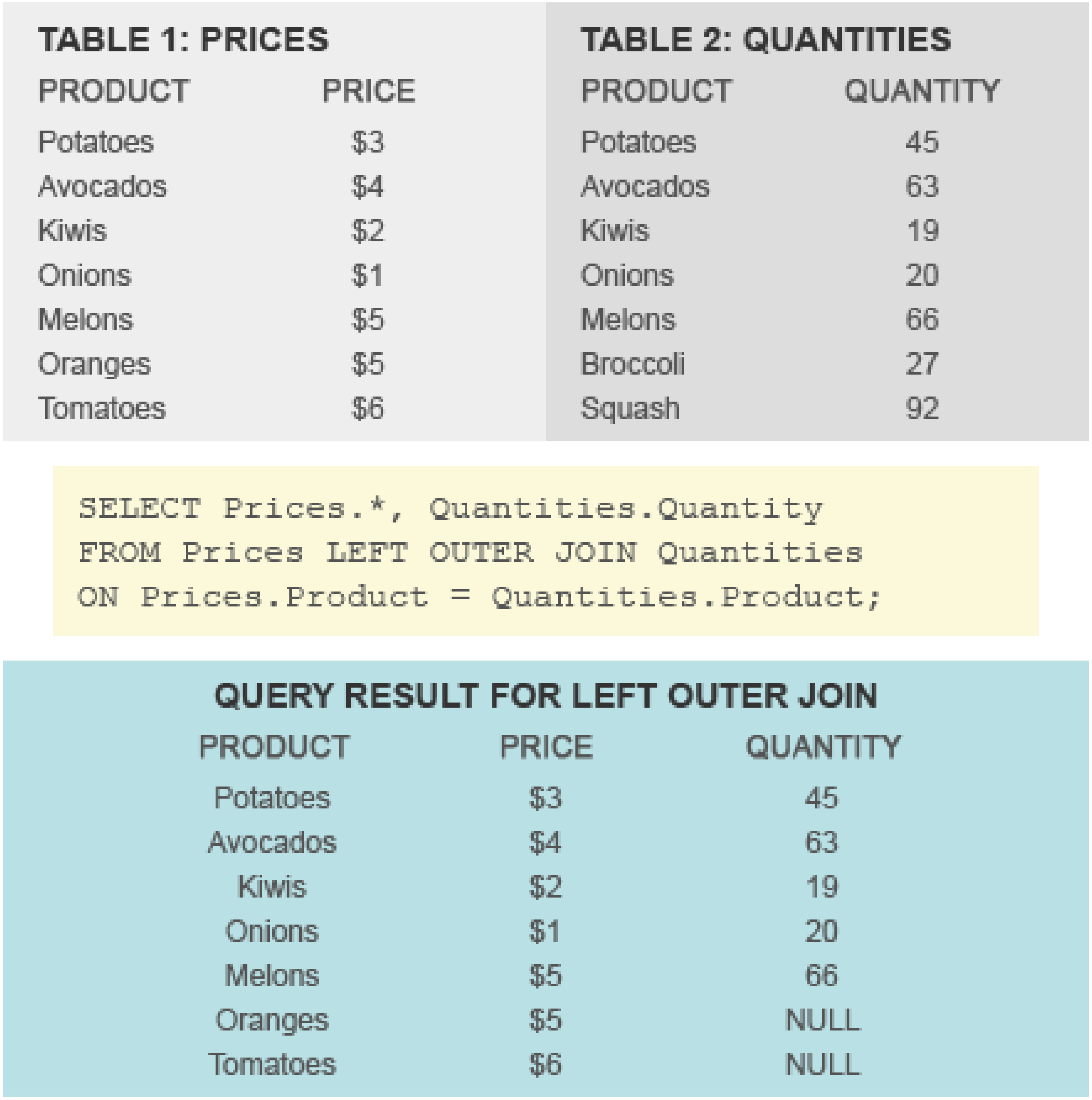


TABLE 1: PRICES

PRODUCT	PRICE
Potatoes	$3
Avocados	$4
Kiwis	$2
Onions	$1
Melons	$5
Oranges	$5
Tomatoes	$6

TABLE 2: QUANTITIES

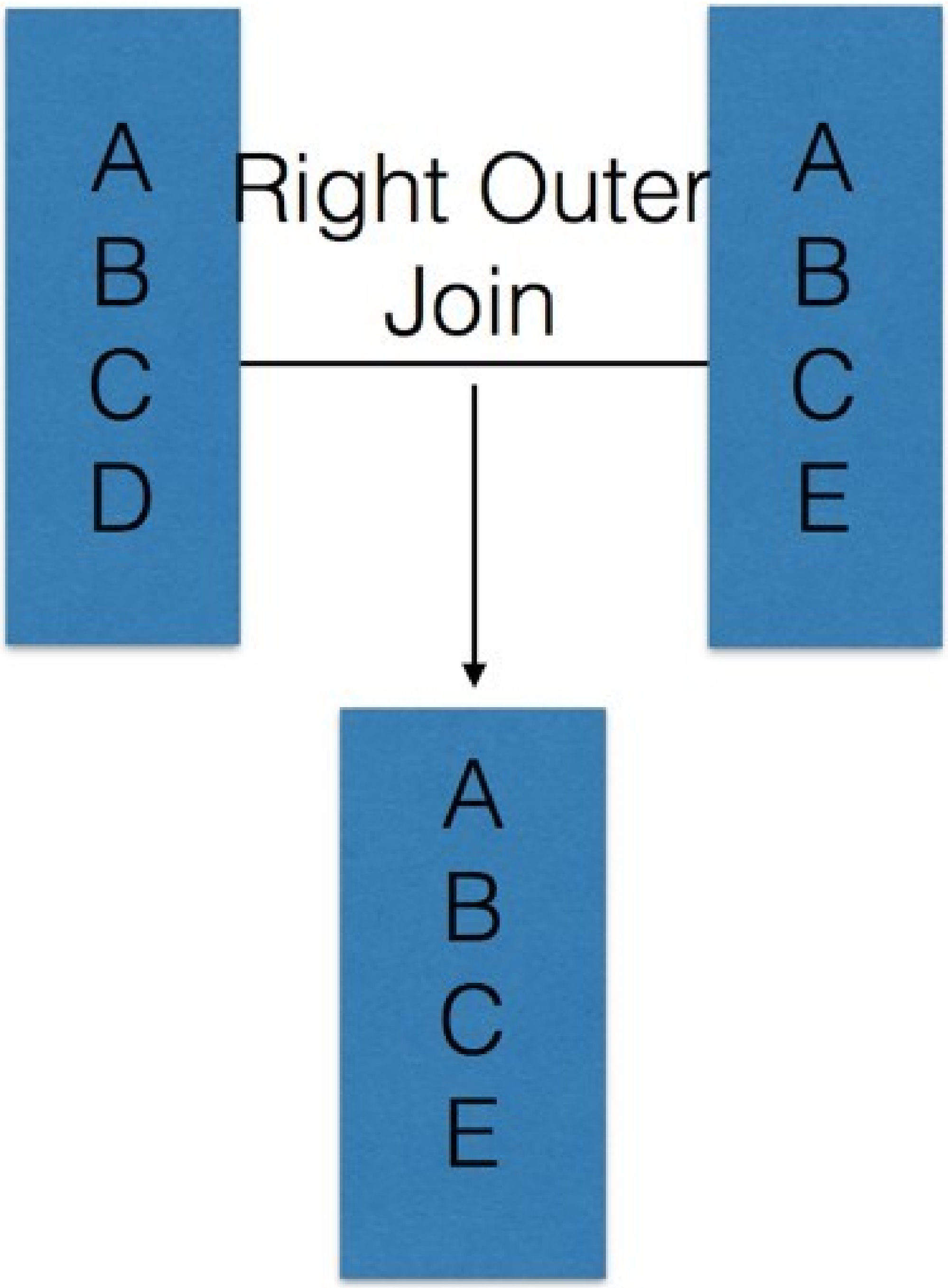
PRODUCT	QUANTITY
Potatoes	45
Avocados	63
Kiwis	19
Onions	20
Melons	66
Broccoli	27
Squash	92

```
SELECT Prices.*, Quantities.Quantity
FROM Prices LEFT OUTER JOIN Quantities
ON Prices.Product = Quantities.Product;
```

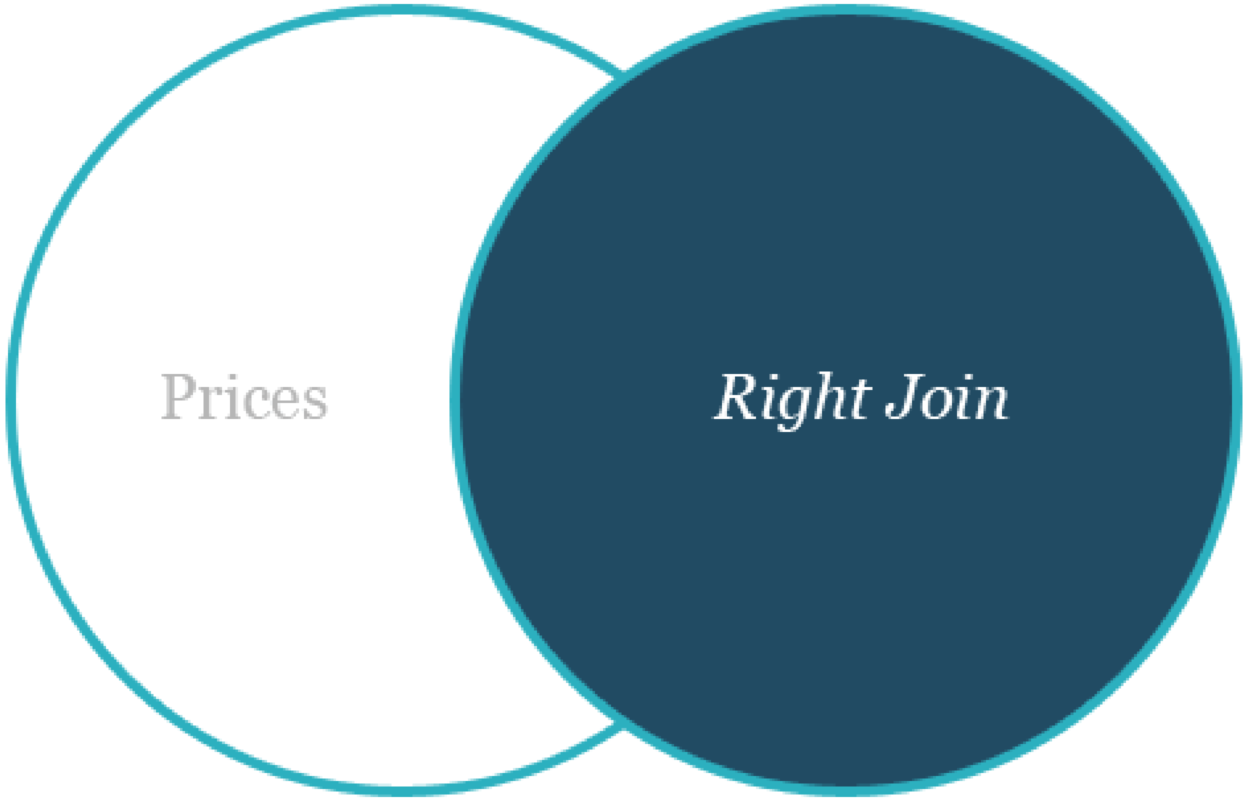
QUERY RESULT FOR LEFT OUTER JOIN

PRODUCT	PRICE	QUANTITY
Potatoes	$3	45
Avocados	$4	63
Kiwis	$2	19
Onions	$1	20
Melons	$5	66
Oranges	$5	NULL
Tomatoes	$6	NULL

Right Outer Joins



Right Outer Join



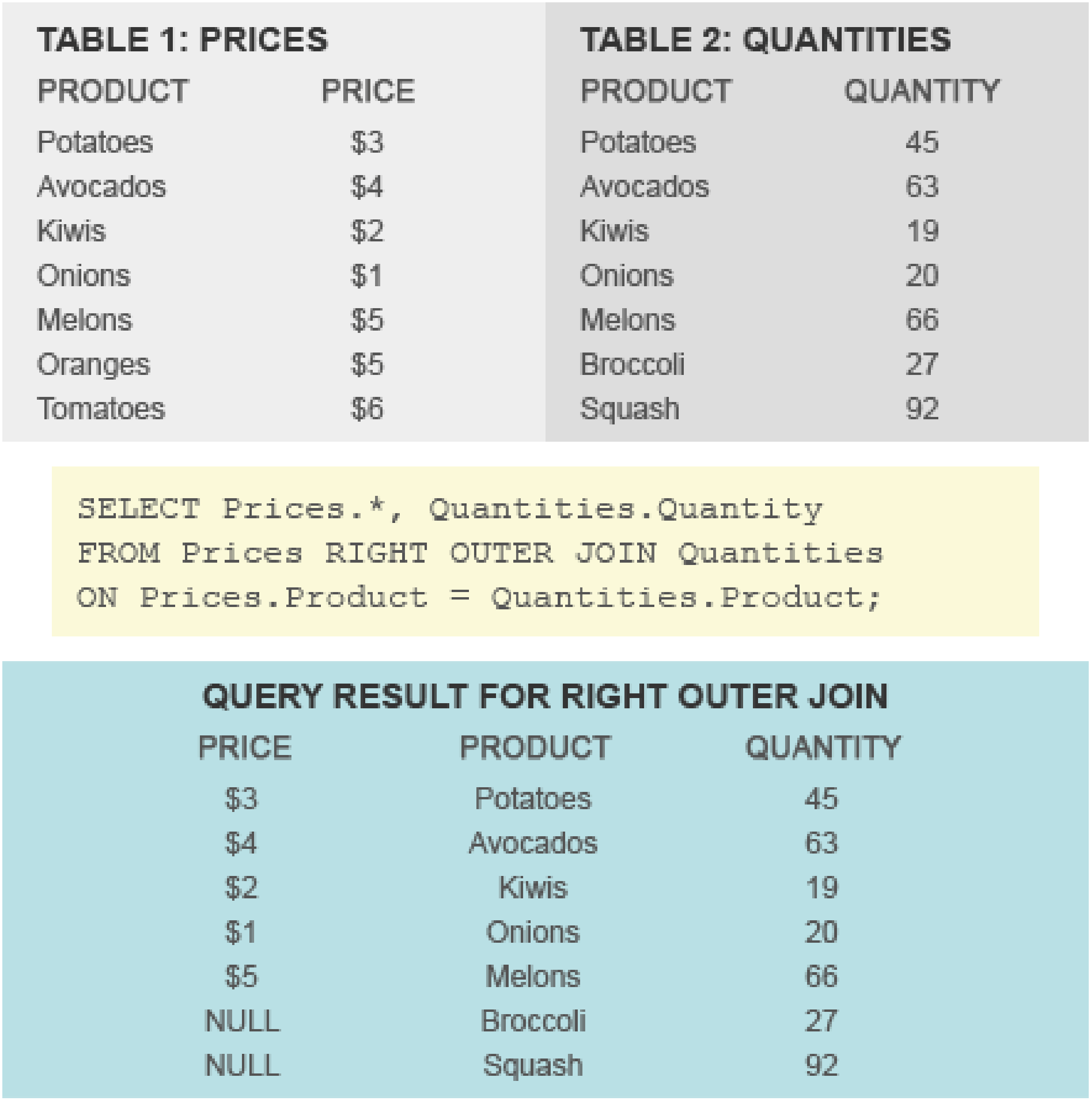


TABLE 1: PRICES

PRODUCT	PRICE
Potatoes	$3
Avocados	$4
Kiwis	$2
Onions	$1
Melons	$5
Oranges	$5
Tomatoes	$6

TABLE 2: QUANTITIES

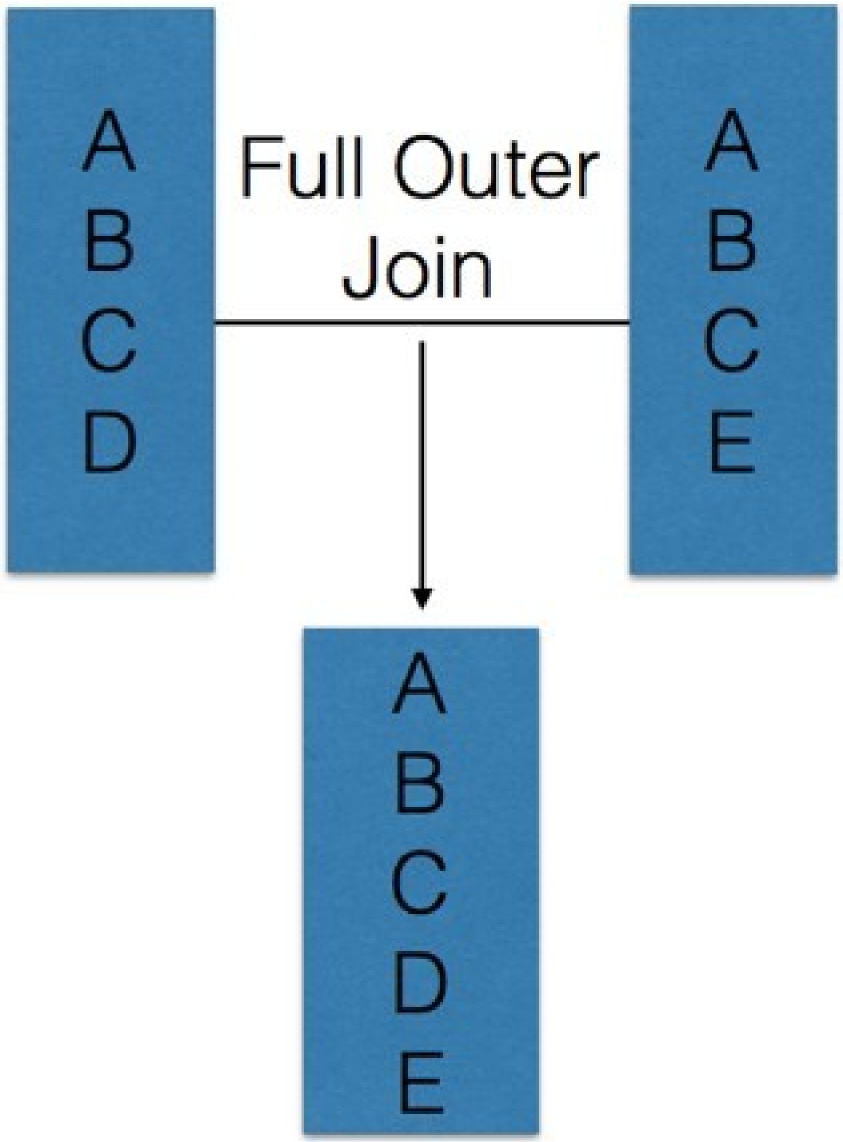
PRODUCT	QUANTITY
Potatoes	45
Avocados	63
Kiwis	19
Onions	20
Melons	66
Broccoli	27
Squash	92

```
SELECT Prices.*, Quantities.Quantity
FROM Prices RIGHT OUTER JOIN Quantities
ON Prices.Product = Quantities.Product;
```

QUERY RESULT FOR RIGHT OUTER JOIN

PRICE	PRODUCT	QUANTITY
$3	Potatoes	45
$4	Avocados	63
$2	Kiwis	19
$1	Onions	20
$5	Melons	66
NULL	Broccoli	27
NULL	Squash	92

Full Outer Joins



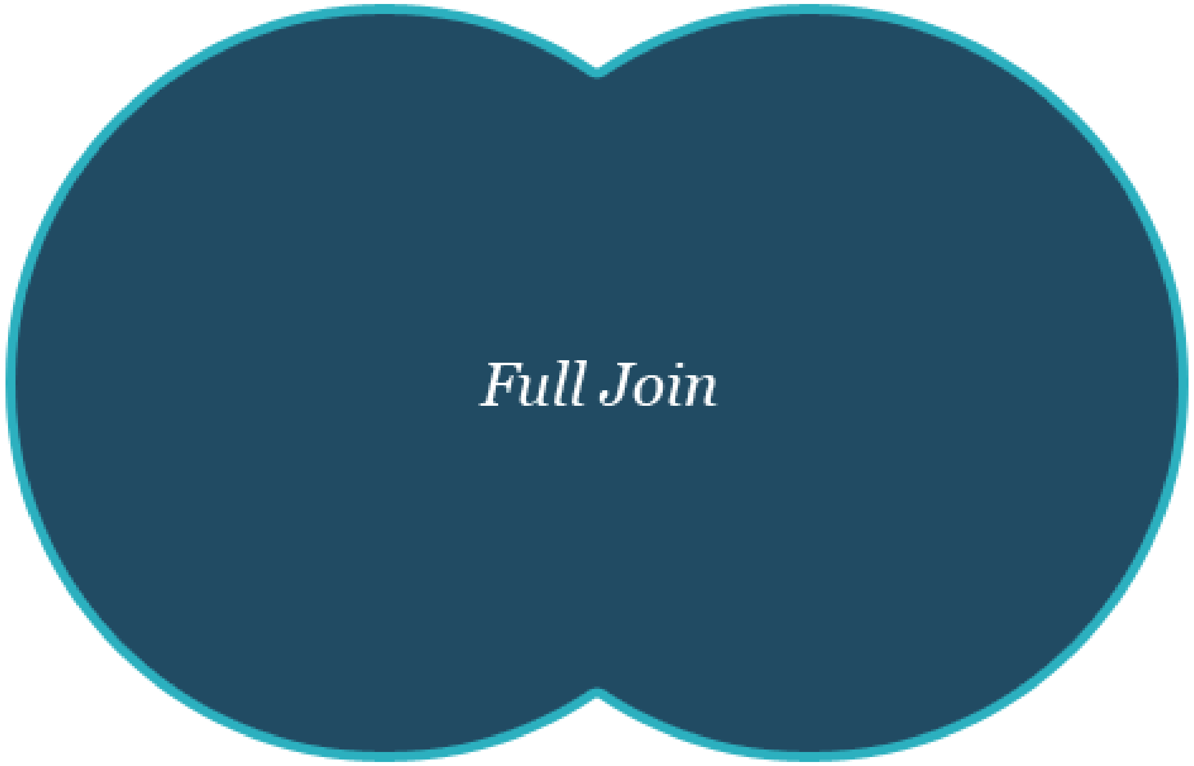
A
B
C
D

Full Outer
Join

A
B
C
E

A
B
C
D
E

Full Outer Join



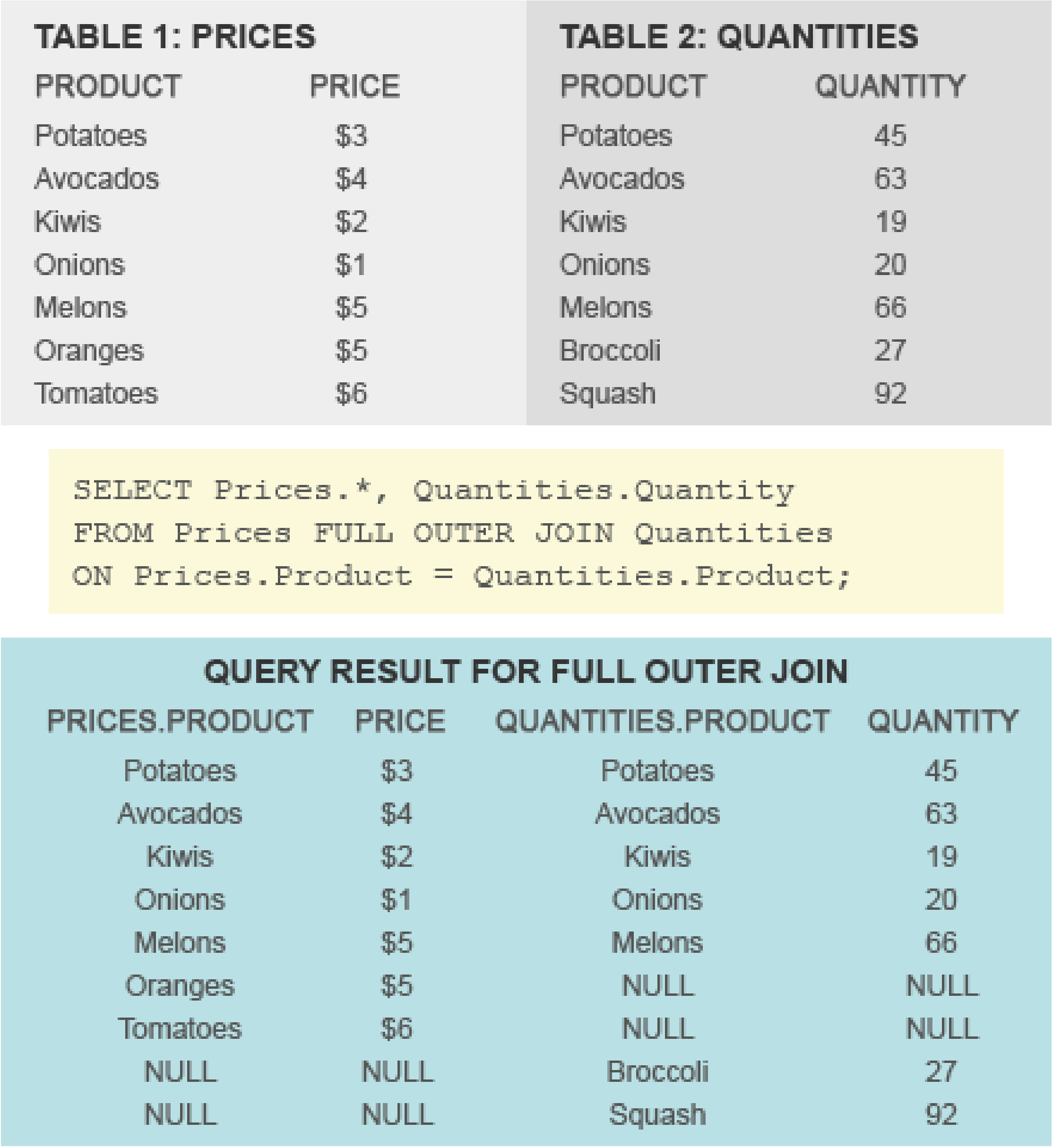


TABLE 1: PRICES

PRODUCT	PRICE
Potatoes	$3
Avocados	$4
Kiwis	$2
Onions	$1
Melons	$5
Oranges	$5
Tomatoes	$6

TABLE 2: QUANTITIES

PRODUCT	QUANTITY
Potatoes	45
Avocados	63
Kiwis	19
Onions	20
Melons	66
Broccoli	27
Squash	92

```
SELECT Prices.*, Quantities.Quantity
FROM Prices FULL OUTER JOIN Quantities
ON Prices.Product = Quantities.Product;
```

QUERY RESULT FOR FULL OUTER JOIN

PRICES.PRODUCT	PRICE	QUANTITIES.PRODUCT	QUANTITY
Potatoes	$3	Potatoes	45
Avocados	$4	Avocados	63
Kiwis	$2	Kiwis	19
Onions	$1	Onions	20
Melons	$5	Melons	66
Oranges	$5	NULL	NULL
Tomatoes	$6	NULL	NULL
NULL	NULL	Broccoli	27
NULL	NULL	Squash	92