JOINS IN SQL

SELF JOIN

INNER JOIN

FULL JOIN

JOINS

OUTER JOIN

CROSS JOIN

INNER JOIN:

An INNER JOIN is used to combine rows from two or more tables based on a related column between them. It only returns the rows where there is a match in both tables.

SYNTAX:

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.common_column = table2.common_column;
```

-- 1. Write a SQL query to find the first name, last name, department, city, and state
-- province for each employee

SOLUTION:

```
select first_name,last_name,d.department_name,city,STATE_PROVINCE
from employee_hr_data e inner join department_hr_data d on
e.DEPARTMENT_ID = d.department_id
join location_hr_data l on l.LOCATION_ID = d.LOCATION_ID ;
```

OUTPUT:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

first_name	last_name	department_name	city	STATE_PROVINCE
Steven	King	Executive	Seattle	Washington
Neena	Kochhar	Executive	Seattle	Washington
Lex	De Haan	Executive	Seattle	Washington
Alexander	Hunold	IT	Southlake	Texas
Bruce	Ernst	IT	Southlake	Texas
David	Austin	IT	Southlake	Texas
Valli	Pataballa	IT	Southlake	Texas
Diana	Lorentz	IT	Southlake	Texas
Nancy	Greenberg	Finance	Seattle	Washington
Daniel	Faviet	Finance	Seattle	Washington
John	Chen	Finance	Seattle	Washington
Ismael	Sciarra	Finance	Seattle	Washington

OUTER JOIN:

An OUTER JOIN is used to retrieve records from two or more tables, including those records that do not have matching rows in the other tables. There are three types of outer joins: **LEFT OUTER JOIN, RIGHT OUTER JOIN, and FULL OUTER JOIN.**

LET’S TALK ABOUT EACH ONE

LEFT OUTER JOIN:

The LEFT OUTER JOIN (or simply LEFT JOIN) returns all rows from the left table and the matched rows from the right table. If no match is found, NULL values are returned for columns from the right table.

SYNTAX:

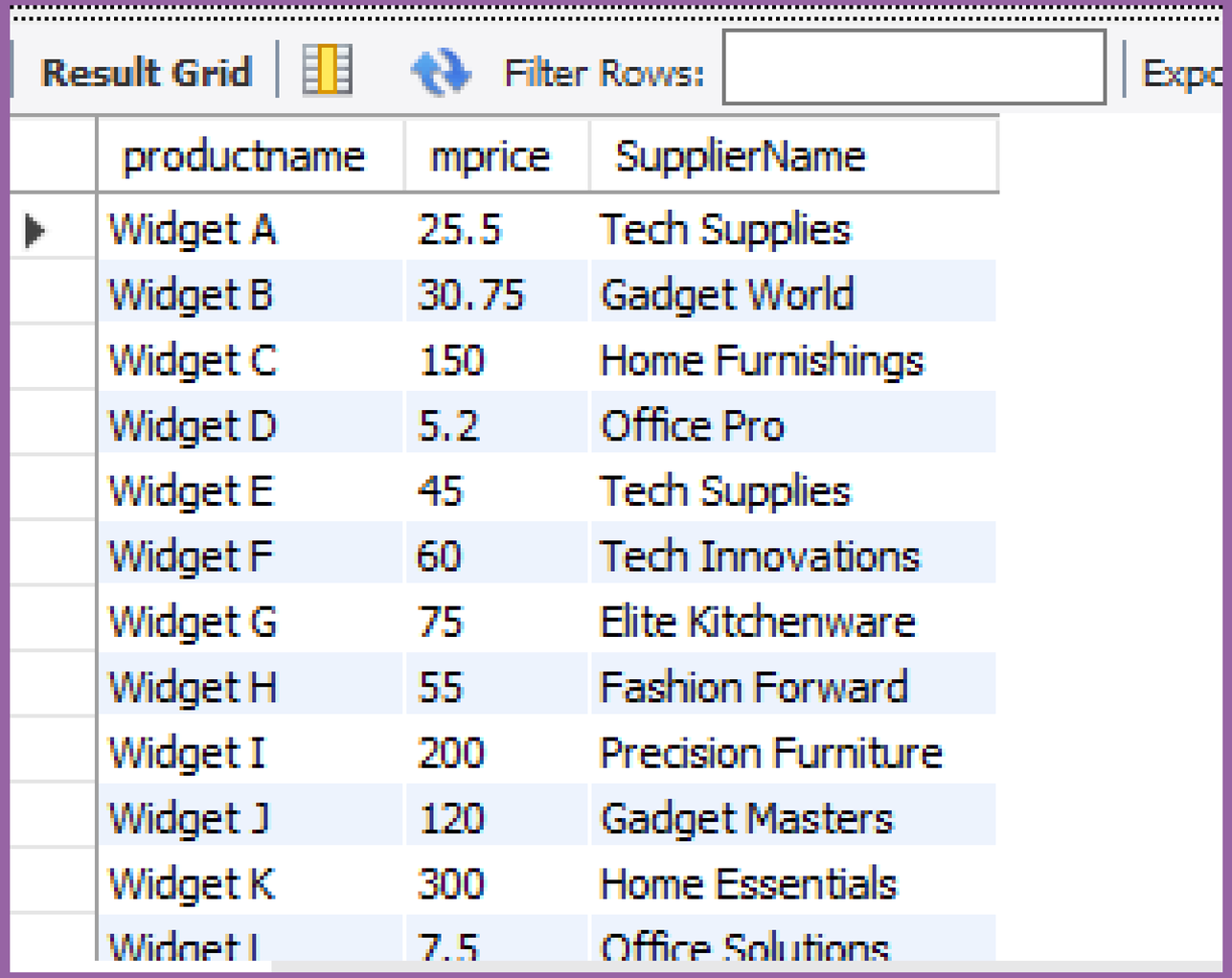
```
SELECT columns
FROM table1
LEFT OUTER JOIN table2
ON table1.common_column = table2.common_column;
```

How can you find the maximum unit price for each product and identify the supplier associated with that maximum price using the products and supplier tables?

SOLUTION:

```
select productname,max(UnitPrice) as mprice,SupplierName from products p
left join supplier s on p.SupplierID=s.SupplierID group by
ProductName,SupplierName;
```

OUTPUT:



Result Grid | Filter Rows: | Expo

productname	mprice	SupplierName
Widget A	25.5	Tech Supplies
Widget B	30.75	Gadget World
Widget C	150	Home Furnishings
Widget D	5.2	Office Pro
Widget E	45	Tech Supplies
Widget F	60	Tech Innovations
Widget G	75	Elite Kitchenware
Widget H	55	Fashion Forward
Widget I	200	Precision Furniture
Widget J	120	Gadget Masters
Widget K	300	Home Essentials
Widget L	7.5	Office Solutions

RIGHT OUTER JOIN:

The RIGHT OUTER JOIN (or simply RIGHT JOIN) returns all rows from the right table and the matched rows from the left table. If no match is found, NULL values are returned for columns from the left table.

SYNTAX:

```
SELECT columns
FROM table1
RIGHT OUTER JOIN table2
ON table1.common_column = table2.common_column;
```

How can you identify the highest unit price for each product along with the supplier details, including cases where products might not have associated suppliers, by using a RIGHT JOIN between the products and supplier tables?

SOLUTION:

```
select productname,max(UnitPrice) as mprice,SupplierName from products p
right join supplier s on p.SupplierID=s.SupplierID group by
ProductName,SupplierName;
```

OUTPUT:

productname	mprice	SupplierName
Widget S	250	Innovate Tech
Widget T	180	Elegant Office Gear
Widget U	40	Stylish Apparel
Widget V	70	NextGen Gadgets
Widget W	55	Home Comforts
Widget X	8	Pro Office Supplies
Widget Y	75	Fashion Line
NULL	NULL	Quality Goods
NULL	NULL	Ultimate Kitchenw…
NULL	NULL	Trendy Electronics
NULL	NULL	Contemporary Fu…

FULL JOIN:

The FULL JOIN returns all rows when there is a match in either table. It combines the results of both LEFT JOIN and RIGHT JOIN. If there is no match, NULL values are returned for columns from the table where no match was found.

SYNTAX:

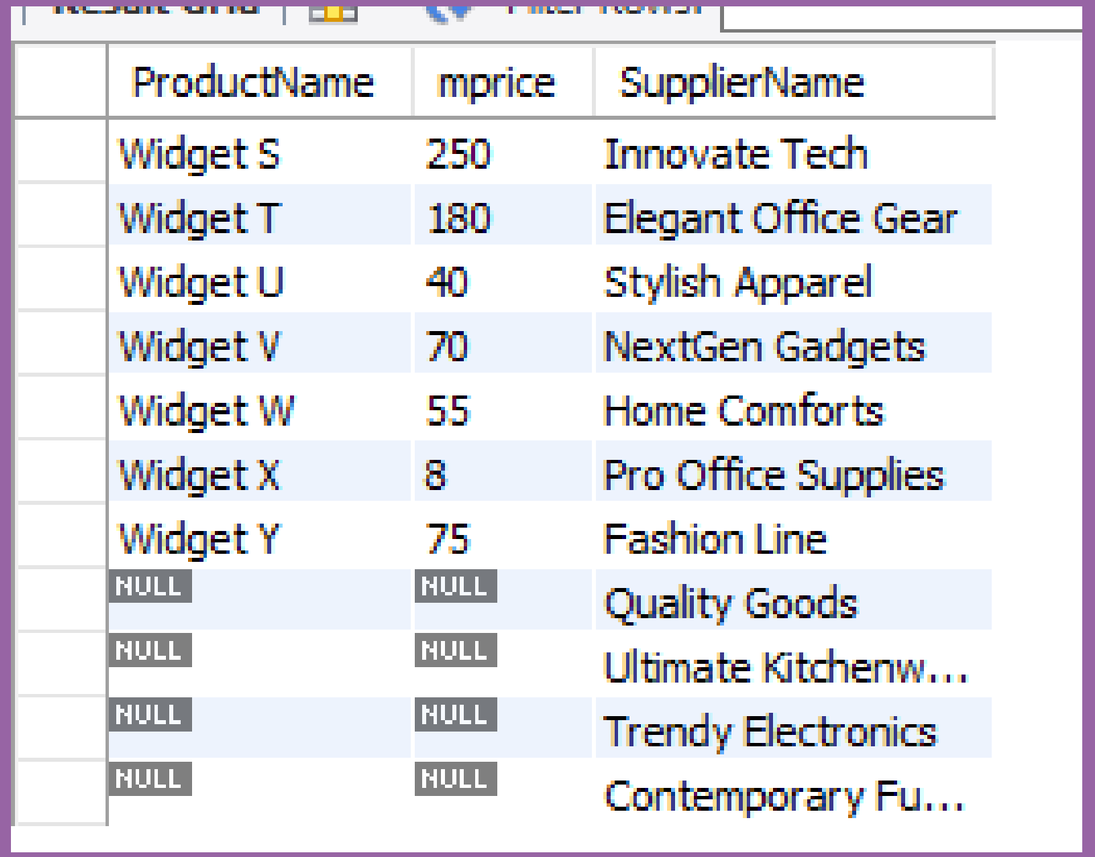
```
SELECT columns
FROM table1
FULL OUTER JOIN table2
ON table1.common_column = table2.common_column;
```

How can you find the maximum unit price for each product and list the associated supplier details, including cases where either products or suppliers might not have corresponding matches, using a FULL JOIN between the products and supplier tables?

SOLUTION:

```
select productname,max(UnitPrice) as mprice,SupplierName from products p full join supplier s on p.SupplierID=s.SupplierID group by ProductName,SupplierName;
```

OUTPUT:



ProductName	mprice	SupplierName
Widget S	250	Innovate Tech
Widget T	180	Elegant Office Gear
Widget U	40	Stylish Apparel
Widget V	70	NextGen Gadgets
Widget W	55	Home Comforts
Widget X	8	Pro Office Supplies
Widget Y	75	Fashion Line
NULL	NULL	Quality Goods
NULL	NULL	Ultimate Kitchenw…
NULL	NULL	Trendy Electronics
NULL	NULL	Contemporary Fu…

SELF JOIN:

A SELF JOIN is a join where a table is joined with itself. This is useful for comparing rows within the same table or querying hierarchical data.

---> When to Use a Self Join:

- Hierarchical Relationships: When dealing with hierarchical data, such as employee-manager relationships.
- Comparative Analysis: When you need to compare rows within the same table.

SYNTAX:

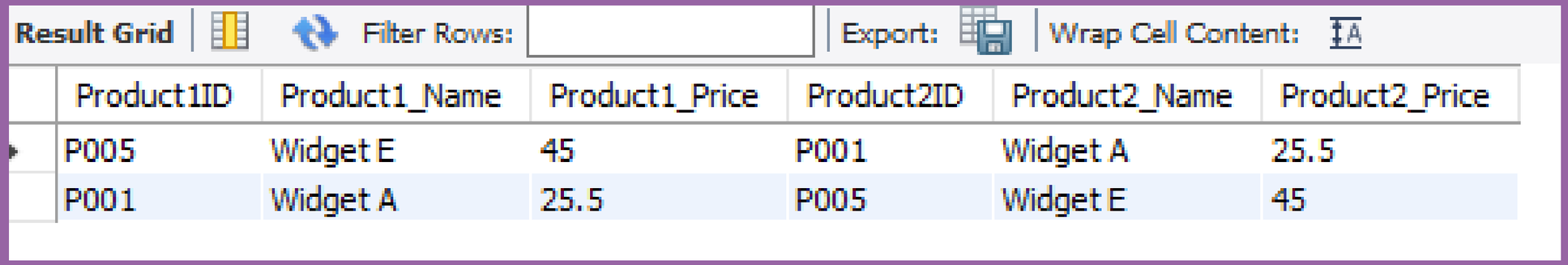
```
SELECT A.columns, B.columns
FROM table A
JOIN table B
ON A.common_column = B.common_column
WHERE condition;
```

Compare Prices of Products from the Same Supplier:

SOLUTION:

```
SELECT p1.ProductID AS Product1_ID, p1.ProductName AS
Product1_Name, p1.UnitPrice AS Product1_Price,
p2.ProductID AS Product2_ID, p2.ProductName AS Product2_Name,
p2.UnitPrice AS Product2_Price
FROM products p1
INNER JOIN products p2
ON p1.SupplierID = p2.SupplierID
AND p1.ProductID <> p2.ProductID;
```

OUTPUT:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Product1ID	Product1_Name	Product1_Price	Product2ID	Product2_Name	Product2_Price
P005	Widget E	45	P001	Widget A	25.5
P001	Widget A	25.5	P005	Widget E	45

CROSS JOIN:

A CROSS JOIN returns the Cartesian product of two tables, which means it combines every row from the first table with every row from the second table. This type of join can produce a very large result set, especially if the tables being joined are large.

SYNTAX:

```
SELECT columns
FROM table1
CROSS JOIN table2;
```

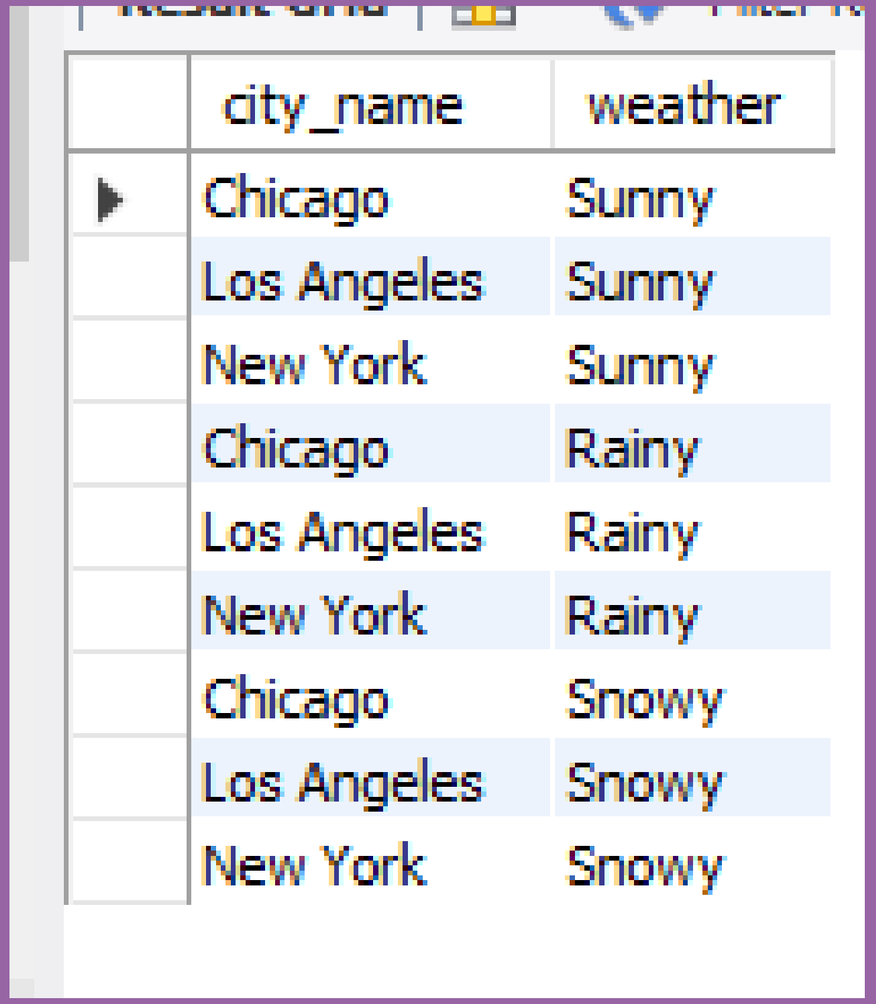
```
CREATE TABLE CITY (
CITY_NAME VARCHAR(30)
);

INSERT INTO CITY VALUES ('NEW YORK');
INSERT INTO CITY VALUES ('LOS ANGELES');
INSERT INTO CITY VALUES ('CHICAGO');
CREATE TABLE WEATHER (
WEATHER VARCHAR(20)
);

INSERT INTO WEATHER VALUES ('SUNNY');
INSERT INTO WEATHER VALUES ('RAINY');
INSERT INTO WEATHER VALUES ('SNOWY');
```

SELECT A.* , B.* FROM CITY A , WEATHER B ;

OUTPUT:



city_name	weather
Chicago	Sunny
Los Angeles	Sunny
New York	Sunny
Chicago	Rainy
Los Angeles	Rainy
New York	Rainy
Chicago	Snowy
Los Angeles	Snowy
New York	Snowy

THANK YOU