Oracle

Data Query Language

- Select
- Join

Select

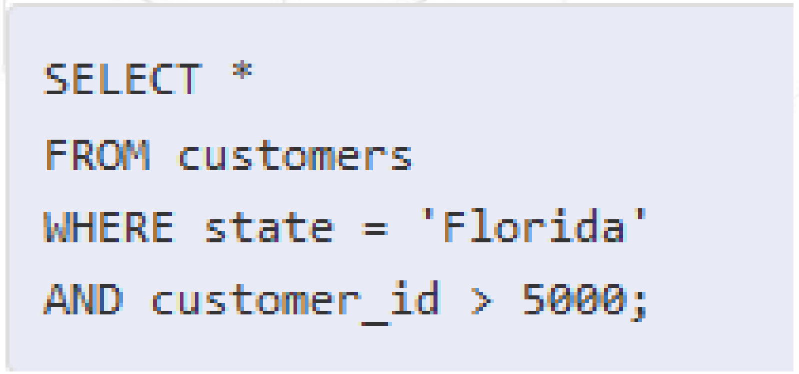
Select

```
SELECT *
FROM homes
WHERE bathrooms >= 2
```

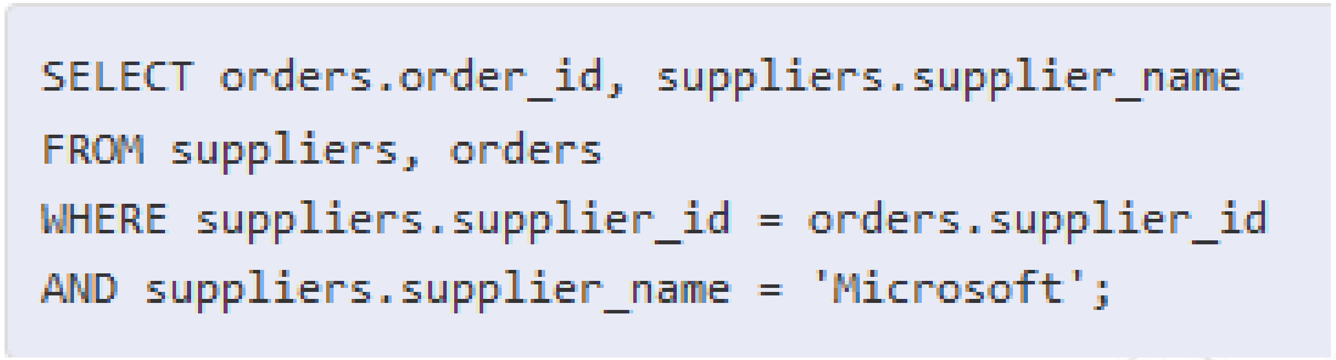
```
SELECT *
FROM homes
WHERE bathrooms >= 2
ORDER BY home_type ASC;
```

```
SELECT home_id, home_type, bathrooms
FROM homes
WHERE home_id < 500
AND home_type = 'two-storey'
ORDER BY home_type ASC, bathrooms DESC;
```

Logical And Operator

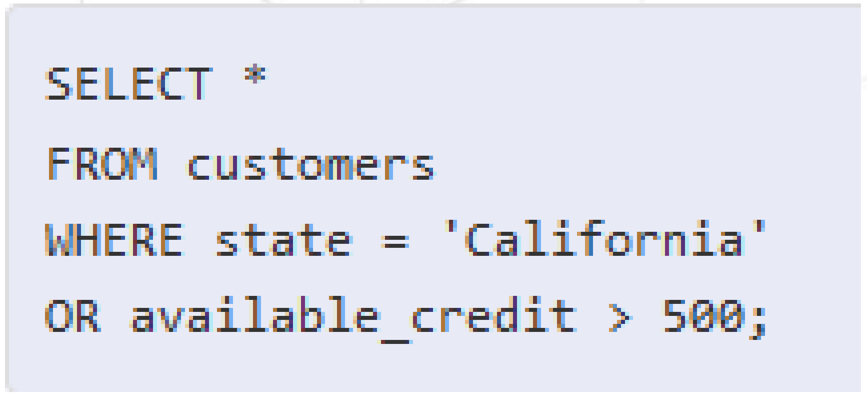


```
SELECT *
FROM customers
WHERE state = 'Florida'
AND customer_id > 5000;
```

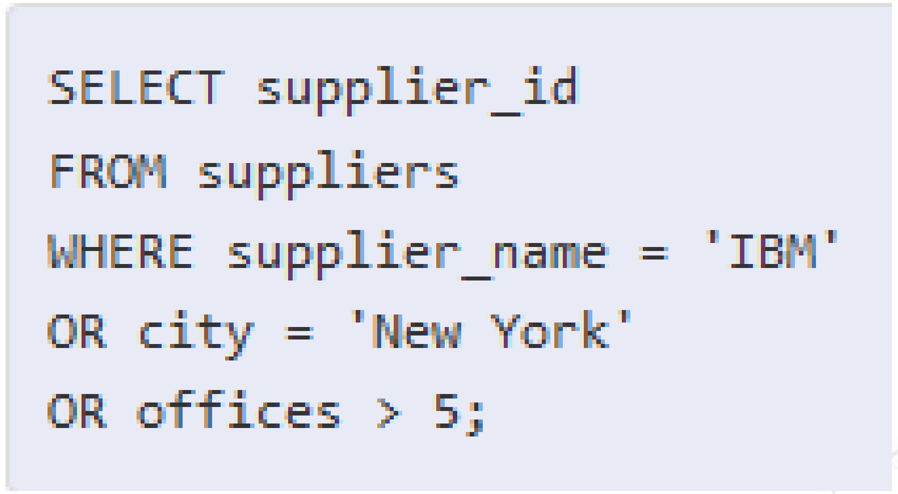


```
SELECT orders.order_id, suppliers.supplier_name
FROM suppliers, orders
WHERE suppliers.supplier_id = orders.supplier_id
AND suppliers.supplier_name = 'Microsoft';
```

Logical OR Operator

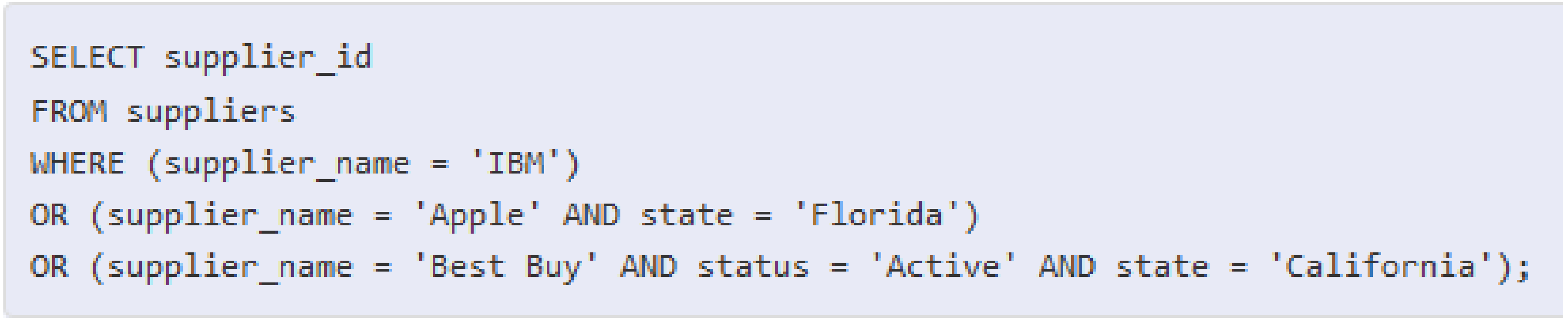


```
SELECT *
FROM customers
WHERE state = 'California'
OR available_credit > 500;
```



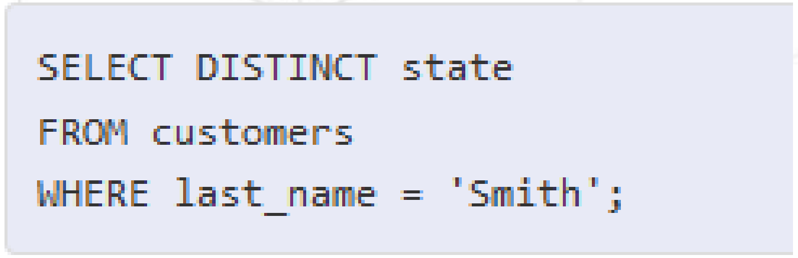
```
SELECT supplier_id
FROM suppliers
WHERE supplier_name = 'IBM'
OR city = 'New York'
OR offices > 5;
```

AND & OR Operator

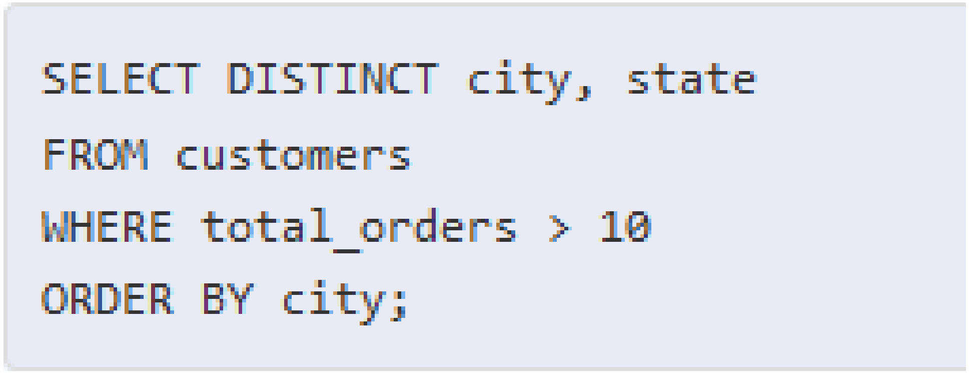


```
SELECT supplier_id
FROM suppliers
WHERE (supplier_name = 'IBM')
OR (supplier_name = 'Apple' AND state = 'Florida')
OR (supplier_name = 'Best Buy' AND status = 'Active' AND state = 'California');
```

Distinct Clause



```
SELECT DISTINCT state
FROM customers
WHERE last_name = 'Smith';
```

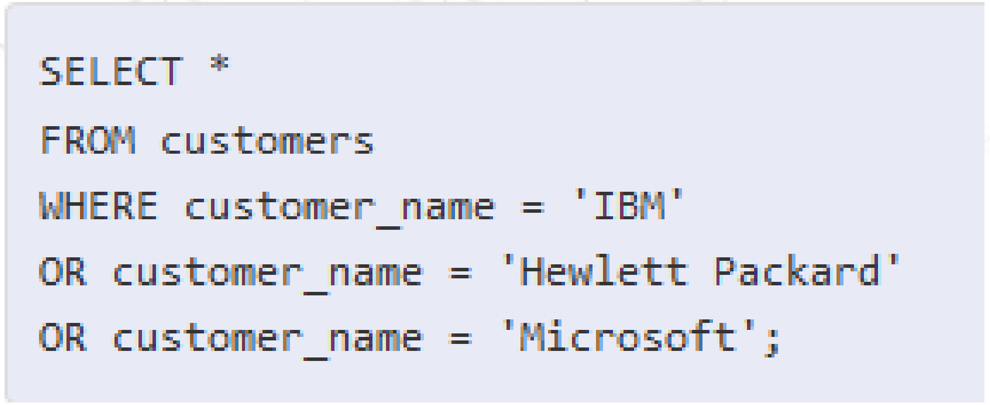


```
SELECT DISTINCT city, state
FROM customers
WHERE total_orders > 10
ORDER BY city;
```

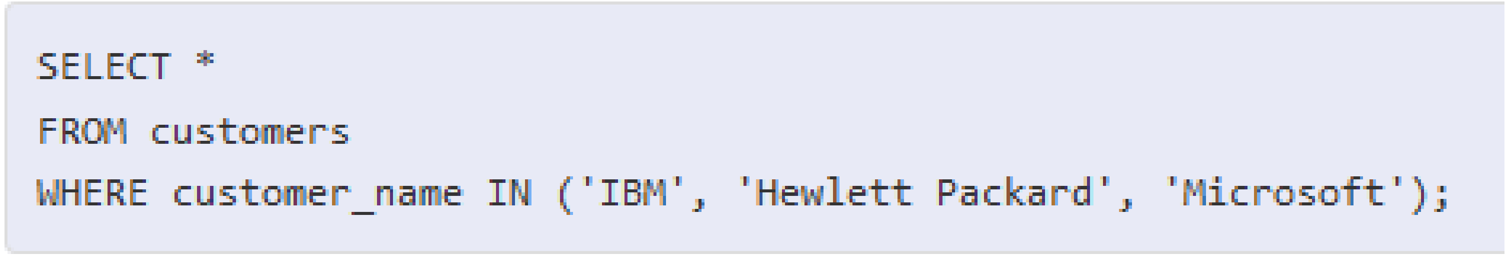
Comparison Operators

Comparison Operator	Description
=	Equal
<>	Not Equal
!=	Not Equal
>	Greater Than
>=	Greater Than or Equal
<	Less Than
<=	Less Than or Equal
IN ( )	Matches a value in a list
NOT	Negates a condition
BETWEEN	Within a range (inclusive)
IS NULL	NULL value
IS NOT NULL	Non-NULL value
LIKE	Pattern matching with % and _
REGEXP_LIKE	Pattern matching with regular expressions
EXISTS	Condition is met if subquery returns at least one row

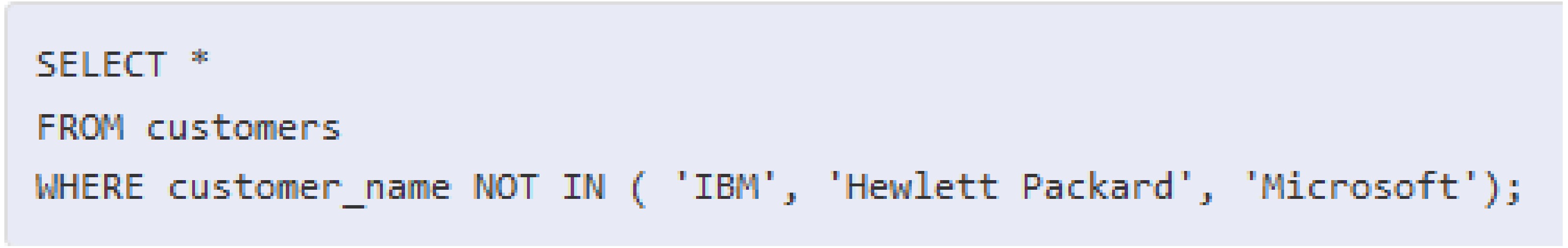
In operator



```
SELECT *
FROM customers
WHERE customer_name = 'IBM'
OR customer_name = 'Hewlett Packard'
OR customer_name = 'Microsoft';
```

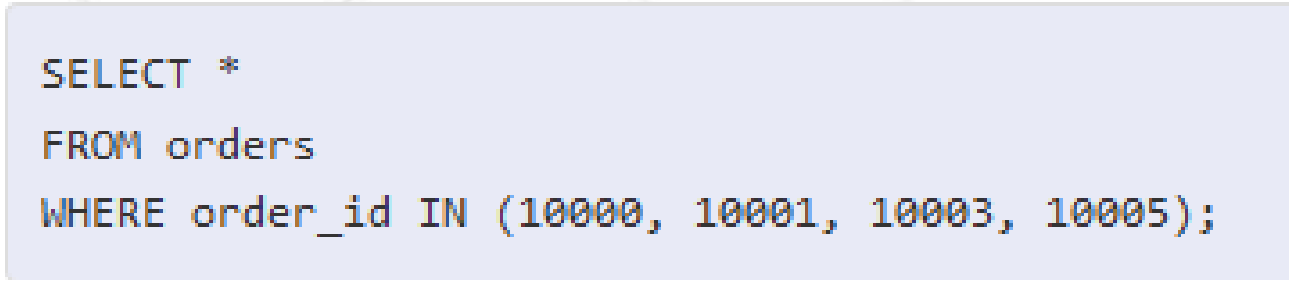


```
SELECT *
FROM customers
WHERE customer_name IN ('IBM', 'Hewlett Packard', 'Microsoft');
```



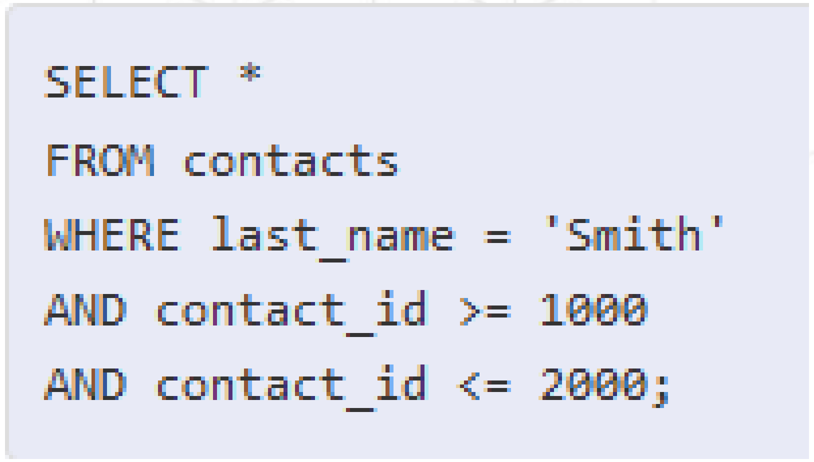
```
SELECT *
FROM customers
WHERE customer_name NOT IN ( 'IBM', 'Hewlett Packard', 'Microsoft');
```

In operator

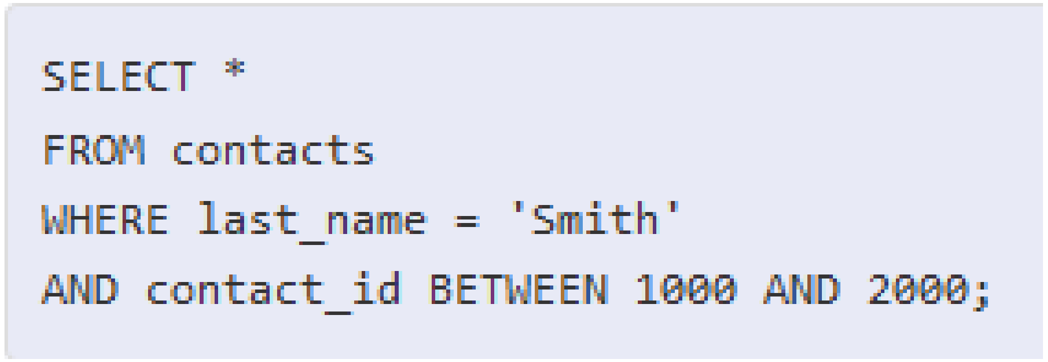


```
SELECT *
FROM orders
WHERE order_id IN (10000, 10001, 10003, 10005);
```

Between Operator



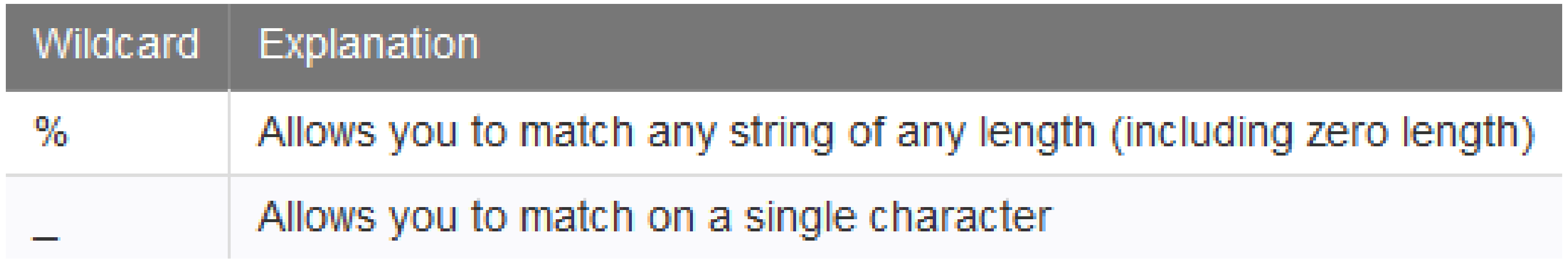
```
SELECT *
FROM contacts
WHERE last_name = 'Smith'
AND contact_id >= 1000
AND contact_id <= 2000;
```



```
SELECT *
FROM contacts
WHERE last_name = 'Smith'
AND contact_id BETWEEN 1000 AND 2000;
```

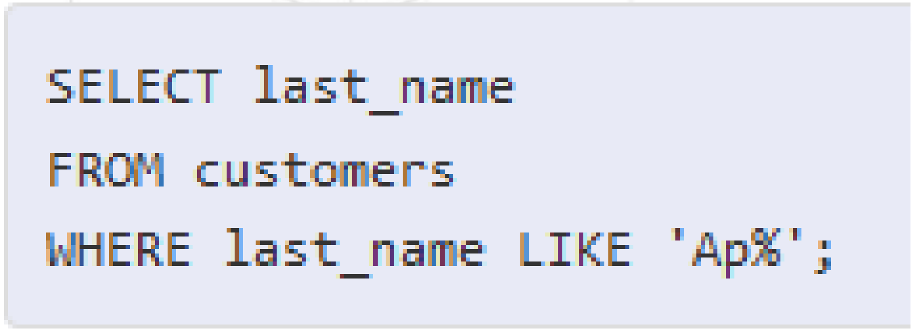
Like Condition

- The Oracle LIKE condition allows wildcards to be used in the WHERE clause of a SELECT, INSERT, UPDATE, or DELETE statement
- To perform pattern matching

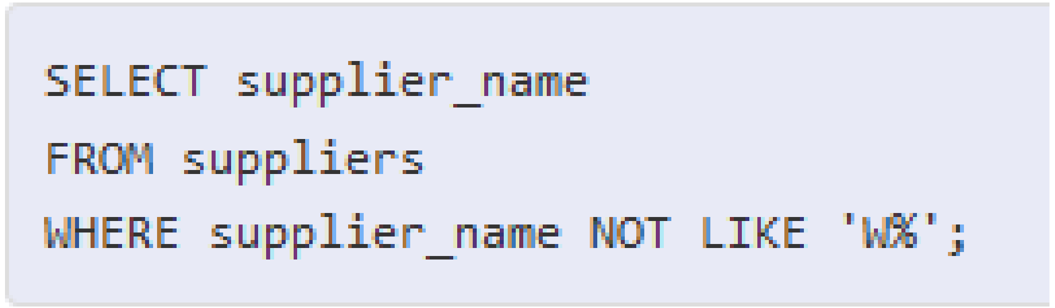


Wildcard	Explanation
%	Allows you to match any string of any length (including zero length)
_	Allows you to match on a single character

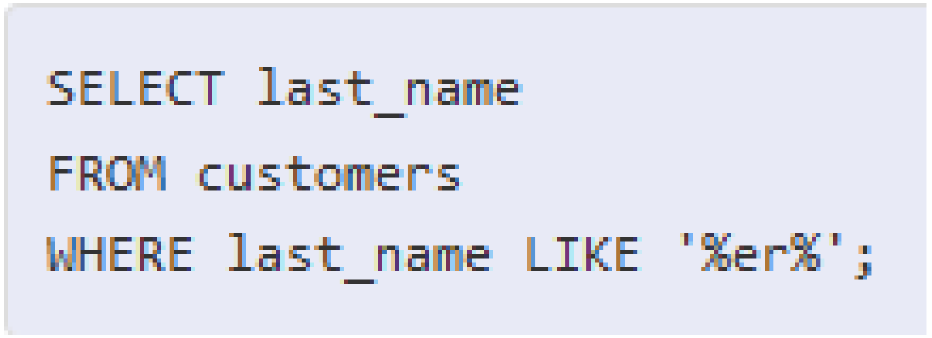
Wildcard Operator - %



```
SELECT last_name
FROM customers
WHERE last_name LIKE 'Ap%';
```

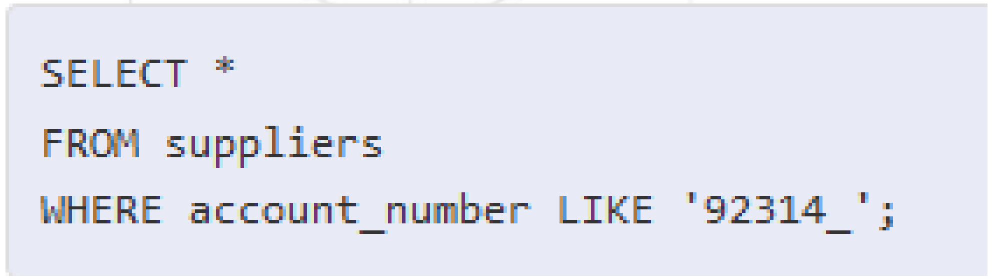


```
SELECT supplier_name
FROM suppliers
WHERE supplier_name NOT LIKE 'W%';
```

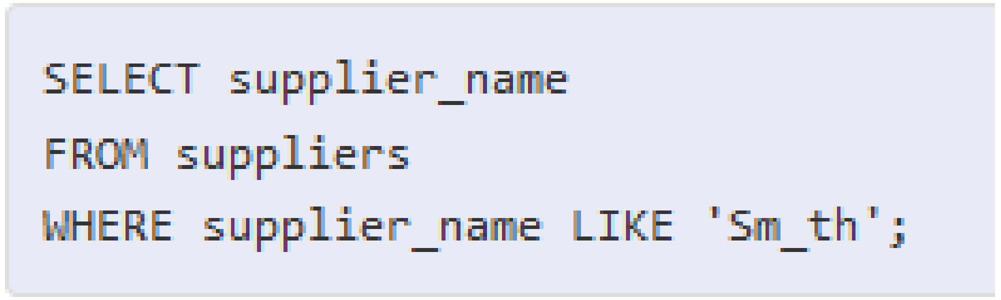


```
SELECT last_name
FROM customers
WHERE last_name LIKE '%er%';
```

Wildcard Operator - _

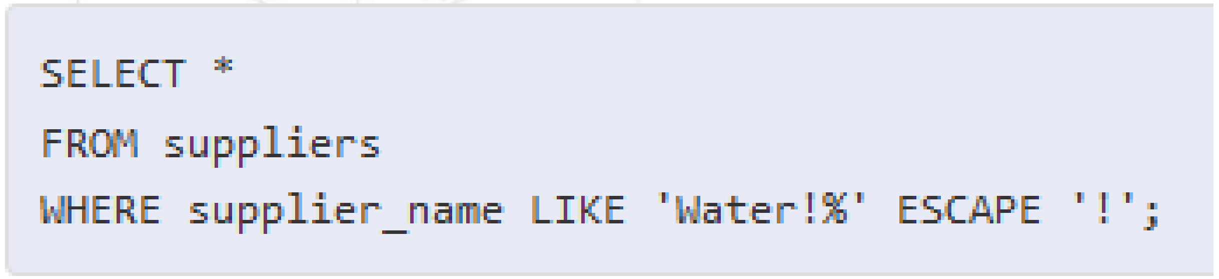


```
SELECT *
FROM suppliers
WHERE account_number LIKE '92314_';
```



```
SELECT supplier_name
FROM suppliers
WHERE supplier_name LIKE 'Sm_th';
```

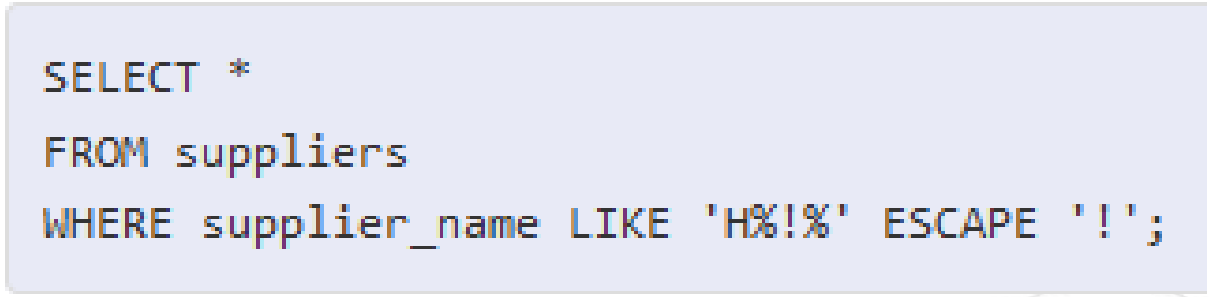
Escape Character



```
SELECT *
FROM suppliers
WHERE supplier_name LIKE 'Water!%' ESCAPE '!';
```

Water%

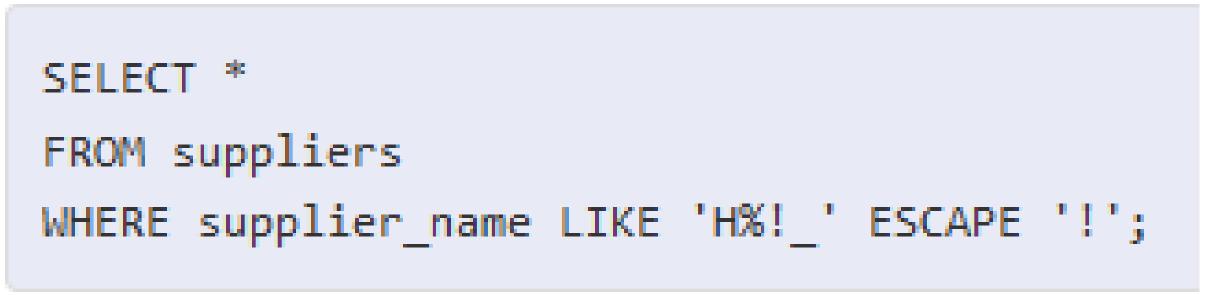
Hello%



```
SELECT *
FROM suppliers
WHERE supplier_name LIKE 'H%!%' ESCAPE '!';
```

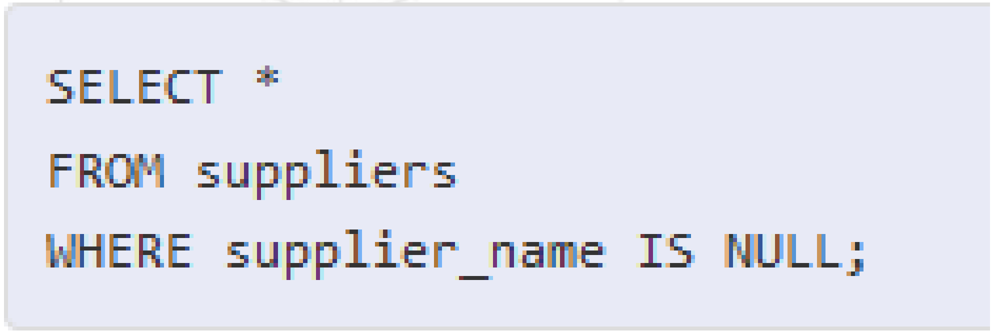
Escape Character

Hello_

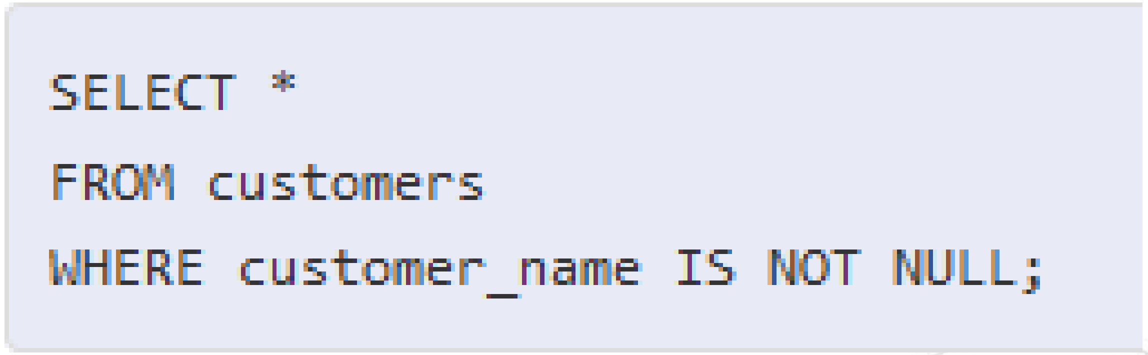


```
SELECT *
FROM suppliers
WHERE supplier_name LIKE 'H%!_' ESCAPE '!';
```

Is / Is NOT NULL operator

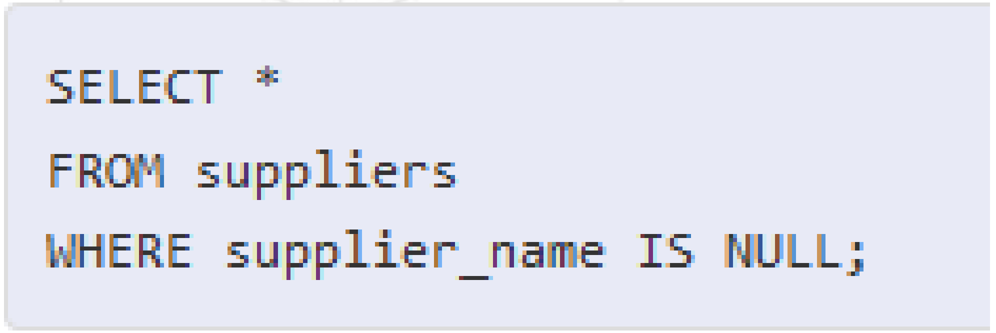


```
SELECT *
FROM suppliers
WHERE supplier_name IS NULL;
```

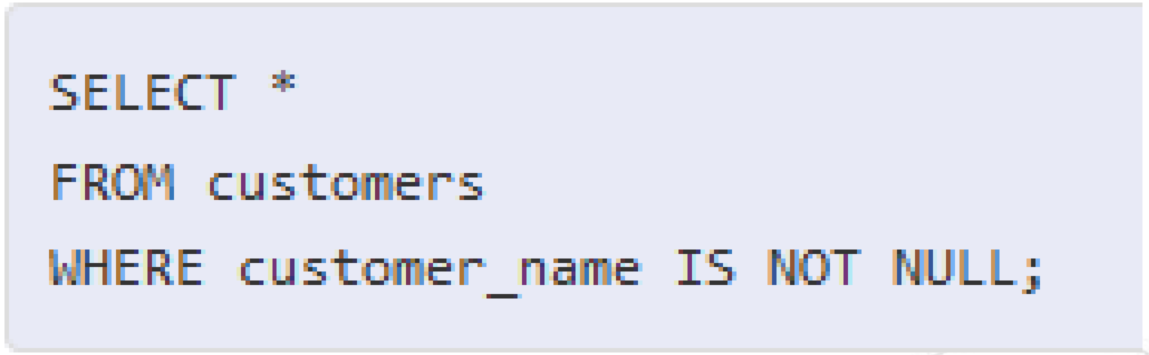


```
SELECT *
FROM customers
WHERE customer_name IS NOT NULL;
```

Is / Is NOT NULL operator



```
SELECT *
FROM suppliers
WHERE supplier_name IS NULL;
```



```
SELECT *
FROM customers
WHERE customer_name IS NOT NULL;
```

REGEXP_LIKE Condition

```
SELECT last_name
FROM contacts
WHERE REGEXP_LIKE (last_name, 'Anders(o|e|a)n');
```

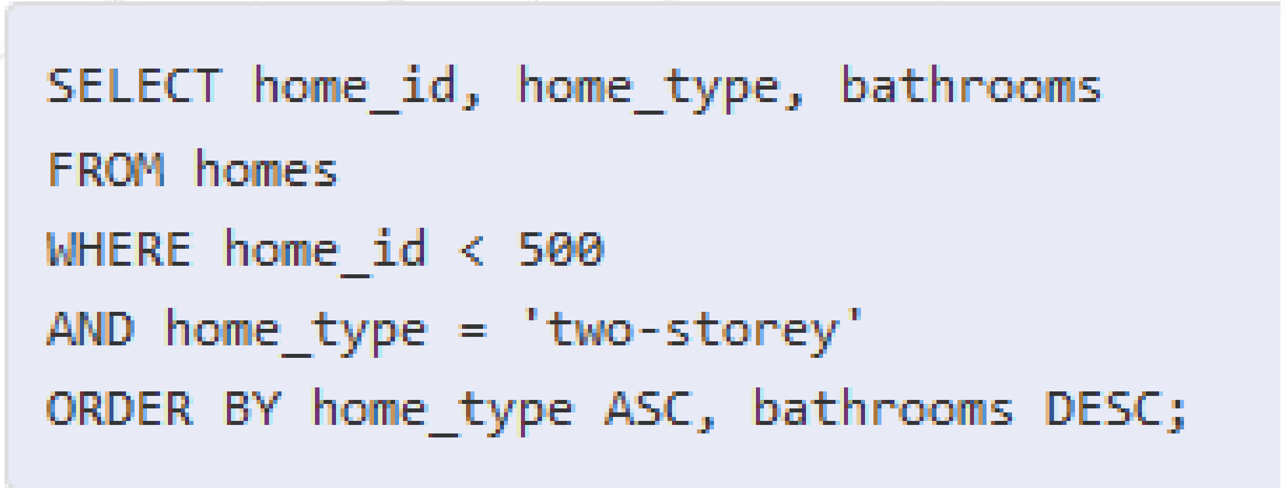
```
SELECT last_name
FROM contacts
WHERE REGEXP_LIKE (last_name, '(*)n$');
```

This REGEXP_LIKE example will return all contacts whose last name ends with 'n'.

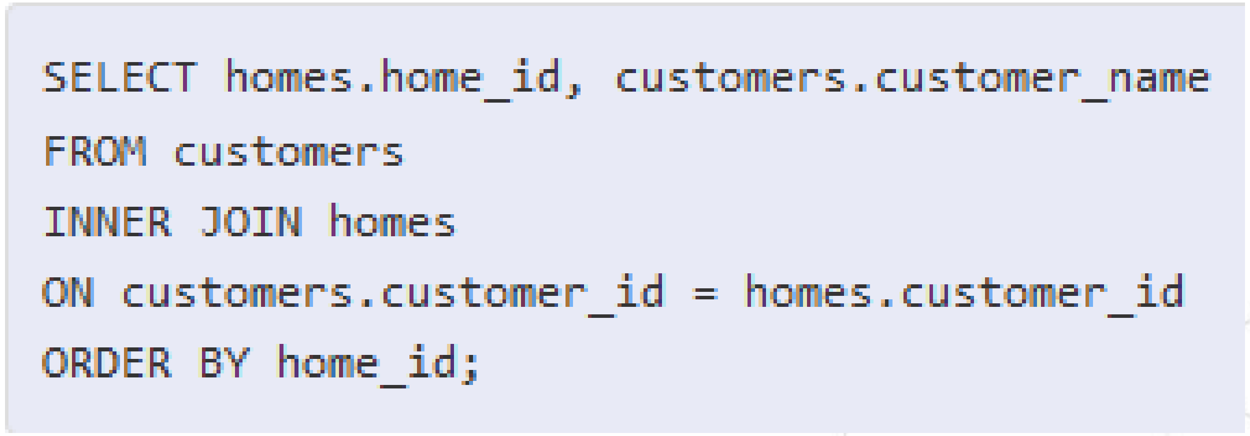
```
SELECT last_name
FROM contacts
WHERE REGEXP_LIKE (last_name, '^A(*)');
```

This REGEXP_LIKE example will return all contacts whose *last name* starts with 'A'

Select



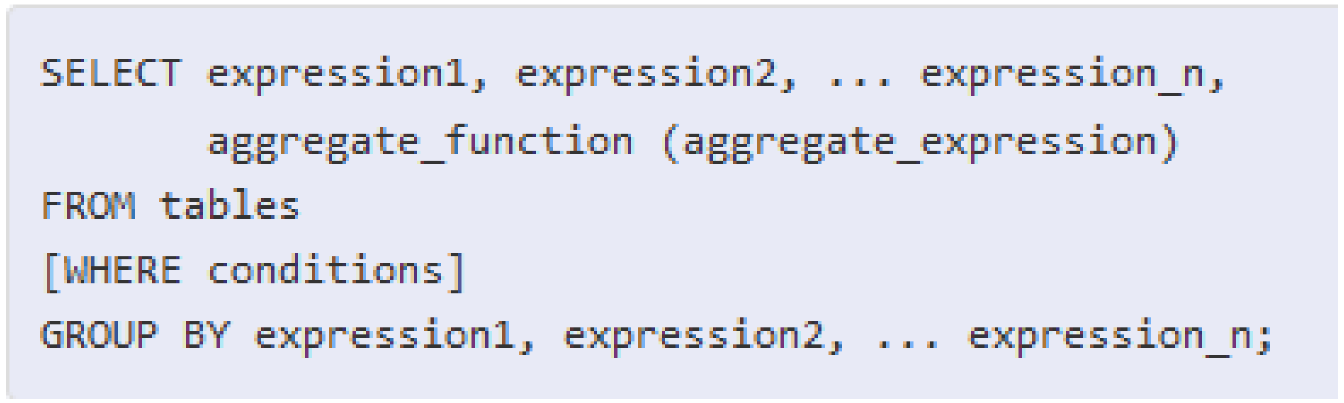
```
SELECT home_id, home_type, bathrooms
FROM homes
WHERE home_id < 500
AND home_type = 'two-storey'
ORDER BY home_type ASC, bathrooms DESC;
```



```
SELECT homes.home_id, customers.customer_name
FROM customers
INNER JOIN homes
ON customers.customer_id = homes.customer_id
ORDER BY home_id;
```

Group By Clause

The Oracle GROUP BY clause is used in a SELECT statement to collect data across multiple records and group the results by one or more columns



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

Group By Clause

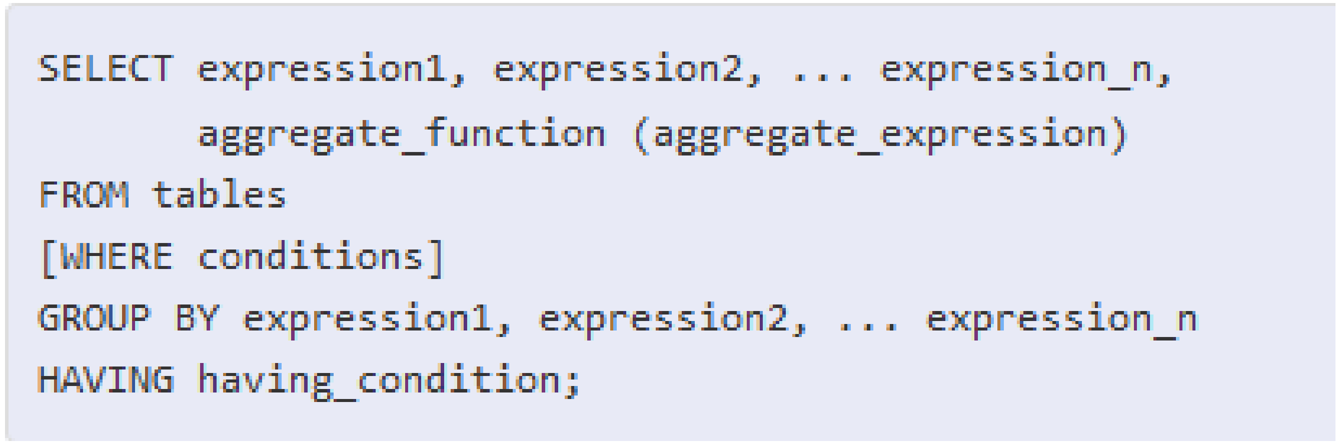
```
SELECT department, MIN(salary) AS "Lowest salary"
FROM employees
GROUP BY department;
```

```
SELECT category, COUNT(*) AS "Number of suppliers"
FROM suppliers
WHERE available_products > 45
GROUP BY category;
```

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

Having Clause

The Oracle 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;
```

Having Clause

```
SELECT department, COUNT(*) AS "Number of employees"
FROM employees
WHERE salary < 49500
GROUP BY department
HAVING COUNT(*) > 10;
```

```
SELECT department, SUM(sales) AS "Total sales"
FROM order_details
GROUP BY department
HAVING SUM(sales) > 25000;
```

Subquery

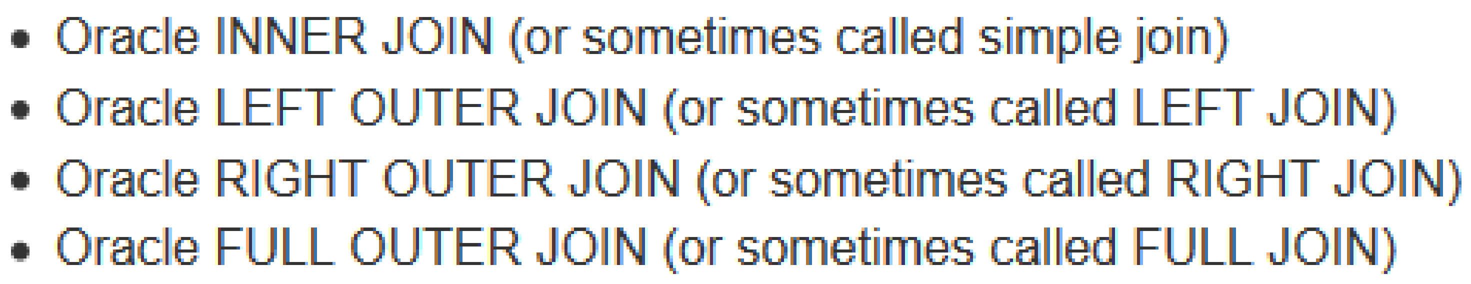
A subquery in

- FROM clause – inline view
- WHERE clause – Nested SubQuery

Joins

Joins

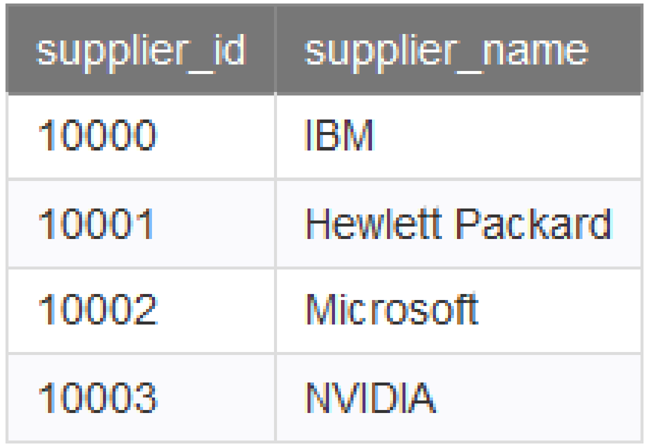
- Oracle JOINS are used to retrieve data from multiple tables
- An Oracle JOIN is performed whenever two or more tables are joined in a SQL statement



- Oracle INNER JOIN (or sometimes called simple join)
- Oracle LEFT OUTER JOIN (or sometimes called LEFT JOIN)
- Oracle RIGHT OUTER JOIN (or sometimes called RIGHT JOIN)
- Oracle FULL OUTER JOIN (or sometimes called FULL JOIN)

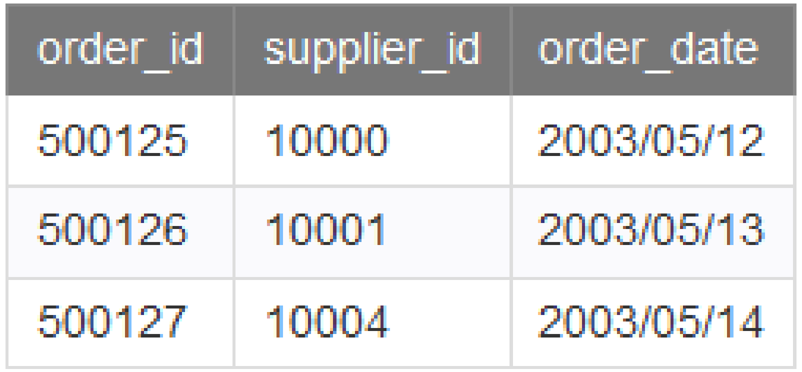
Problem Statement

Supplier



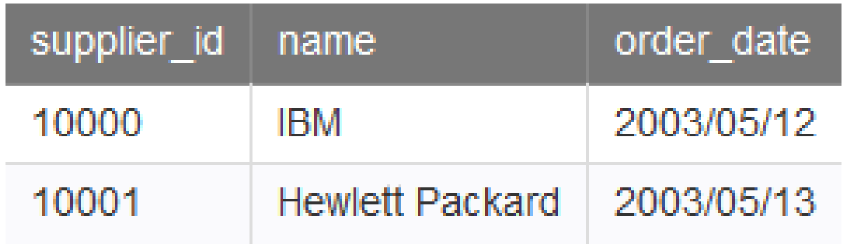
supplier_id	supplier_name
10000	IBM
10001	Hewlett Packard
10002	Microsoft
10003	NVIDIA

Orders



order_id	supplier_id	order_date
500125	10000	2003/05/12
500126	10001	2003/05/13
500127	10004	2003/05/14

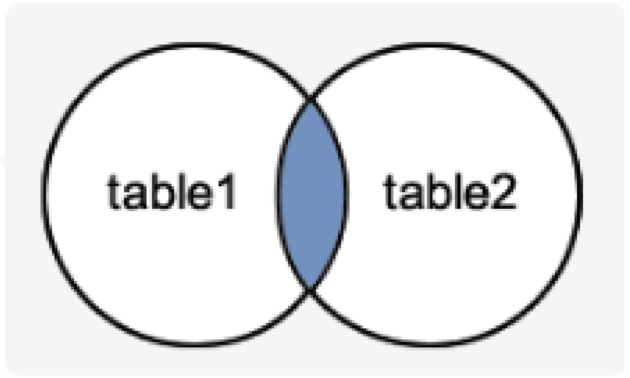
Expected Result



supplier_id	name	order_date
10000	IBM	2003/05/12
10001	Hewlett Packard	2003/05/13

Inner Join

supplier_id	supplier_name
10000	IBM
10001	Hewlett Packard
10002	Microsoft
10003	NVIDIA

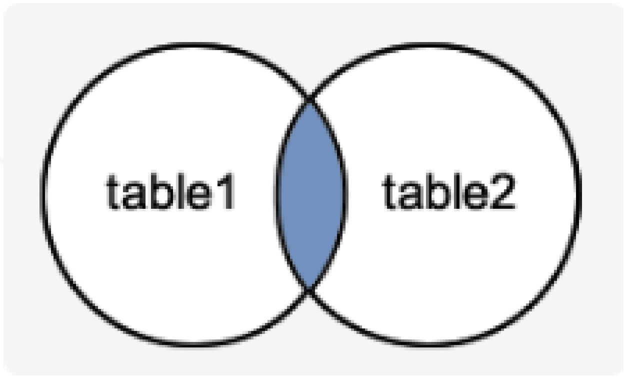


order_id	supplier_id	order_date
500125	10000	2003/05/12
500126	10001	2003/05/13
500127	10004	2003/05/14

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

Inner Join

supplier_id	supplier_name
10000	IBM
10001	Hewlett Packard
10002	Microsoft
10003	NVIDIA



order_id	supplier_id	order_date
500125	10000	2003/05/12
500126	10001	2003/05/13
500127	10004	2003/05/14

```
SELECT suppliers.supplier_id, suppliers.supplier_name, orders.order_date
FROM suppliers
INNER JOIN orders
ON suppliers.supplier_id = orders.supplier_id;
```

Problem Statement

Supplier

supplier_id	supplier_name
10000	IBM
10001	Hewlett Packard
10002	Microsoft
10003	NVIDIA

Orders

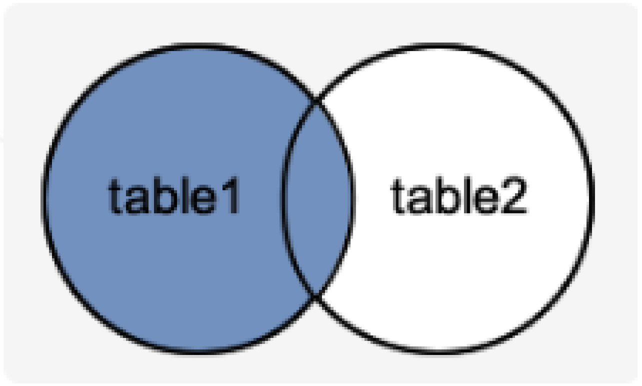
order_id	supplier_id	order_date
500125	10000	2003/05/12
500126	10001	2003/05/13
500127	10004	2003/05/14

Expected Result

supplier_id	supplier_name	order_date
10000	IBM	2003/05/12
10001	Hewlett Packard	2003/05/13
10002	Microsoft	<null>
10003	NVIDIA	<null>

Left [Outer] Join

supplier_id	supplier_name
10000	IBM
10001	Hewlett Packard
10002	Microsoft
10003	NVIDIA

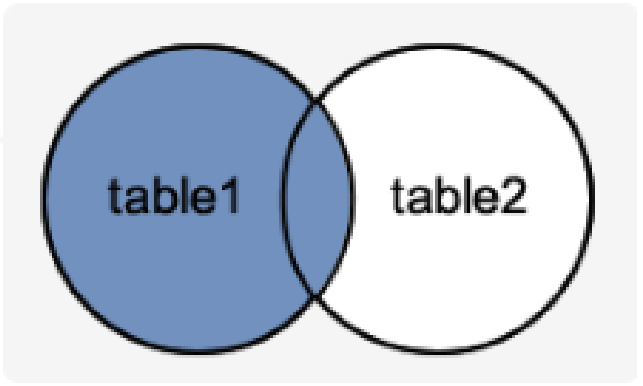


order_id	supplier_id	order_date
500125	10000	2003/05/12
500126	10001	2003/05/13
500127	10004	2003/05/14

```
SELECT columns
FROM table1
LEFT [OUTER] JOIN table2
ON table1.column = table2.column;
```

Left [Outer]Join

supplier_id	supplier_name
10000	IBM
10001	Hewlett Packard
10002	Microsoft
10003	NVIDIA

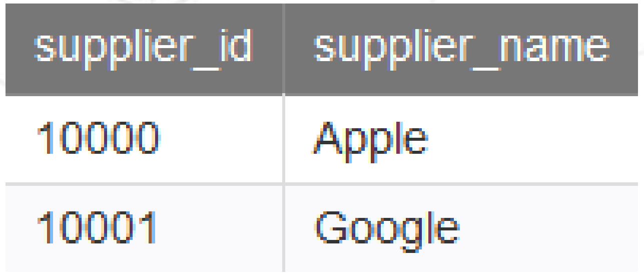


order_id	supplier_id	order_date
500125	10000	2003/05/12
500126	10001	2003/05/13
500127	10004	2003/05/14

```
SELECT suppliers.supplier_id, suppliers.supplier_name, orders.order_date
FROM suppliers
LEFT OUTER JOIN orders
ON suppliers.supplier_id = orders.supplier_id;
```

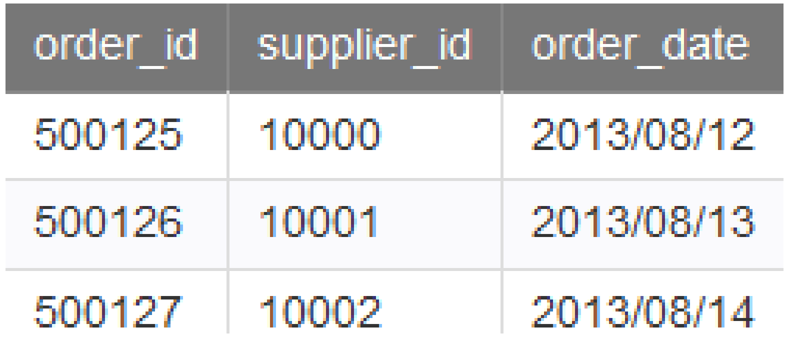
Problem Statement

Supplier



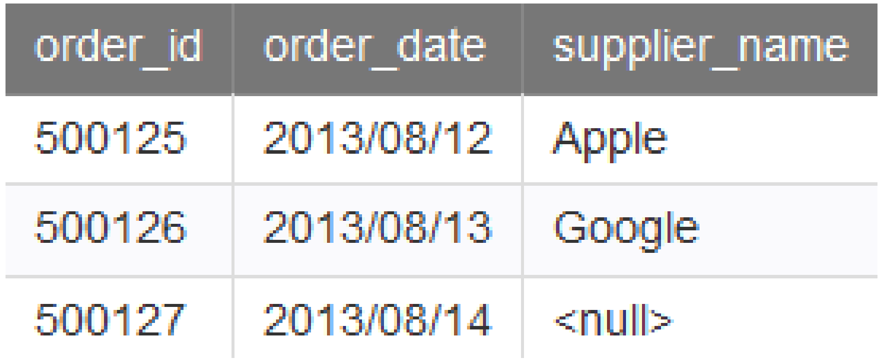
supplier_id	supplier_name
10000	Apple
10001	Google

Orders



order_id	supplier_id	order_date
500125	10000	2013/08/12
500126	10001	2013/08/13
500127	10002	2013/08/14

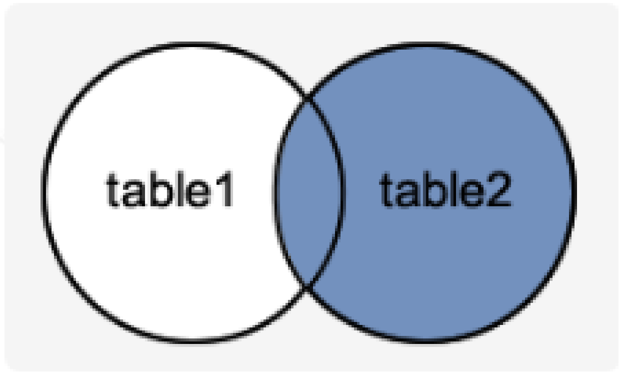
Expected Result



order_id	order_date	supplier_name
500125	2013/08/12	Apple
500126	2013/08/13	Google
500127	2013/08/14	<null>

Right [Outer] Join

supplier_id	supplier_name
10000	Apple
10001	Google

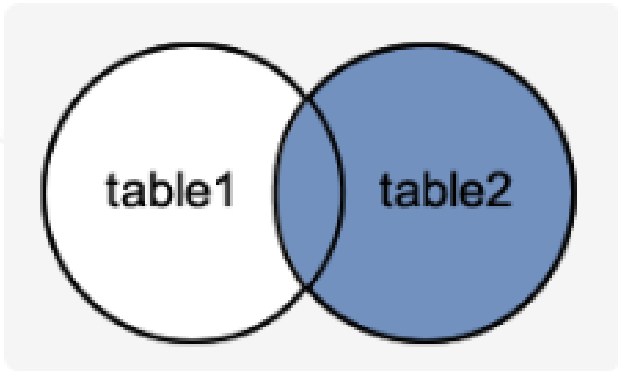


order_id	supplier_id	order_date
500125	10000	2013/08/12
500126	10001	2013/08/13
500127	10002	2013/08/14

```
SELECT columns
FROM table1
RIGHT [OUTER] JOIN table2
ON table1.column = table2.column;
```

Right[Outer]Join

supplier_id	supplier_name
10000	Apple
10001	Google



order_id	supplier_id	order_date
500125	10000	2013/08/12
500126	10001	2013/08/13
500127	10002	2013/08/14

```
SELECT orders.order_id, orders.order_date, suppliers.supplier_name
FROM suppliers
RIGHT OUTER JOIN orders
ON suppliers.supplier_id = orders.supplier_id;
```

Problem Statement

Supplier

supplier_id	supplier_name
10000	IBM
10001	Hewlett Packard
10002	Microsoft
10003	NVIDIA

Orders

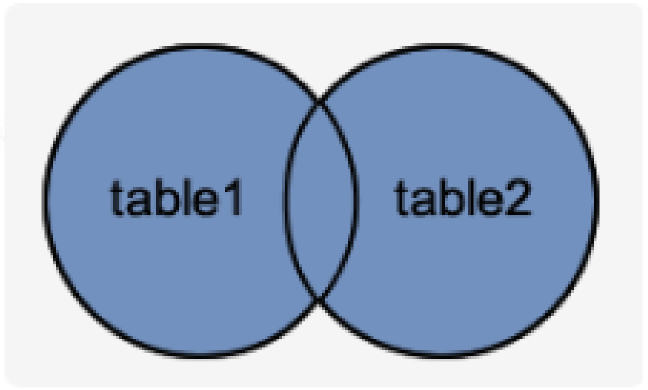
order_id	supplier_id	order_date
500125	10000	2003/05/12
500126	10001	2003/05/13
500127	10004	2003/05/14

Expected Result

supplier_id	supplier_name	order_date
10000	IBM	2013/08/12
10001	Hewlett Packard	2013/08/13
10002	Microsoft	<null>
10003	NVIDIA	<null>
<null>	<null>	2013/08/14

Full [Outer] Join

supplier_id	supplier_name
10000	IBM
10001	Hewlett Packard
10002	Microsoft
10003	NVIDIA

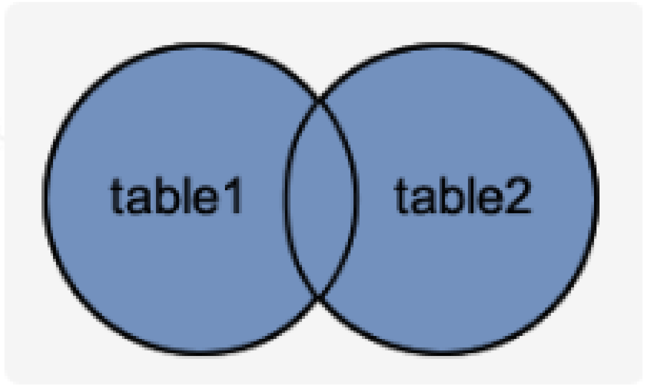


order_id	supplier_id	order_date
500125	10000	2003/05/12
500126	10001	2003/05/13
500127	10004	2003/05/14

```
SELECT columns
FROM table1
FULL [OUTER] JOIN table2
ON table1.column = table2.column;
```

Full [Outer]Join

supplier_id	supplier_name
10000	IBM
10001	Hewlett Packard
10002	Microsoft
10003	NVIDIA



order_id	supplier_id	order_date
500125	10000	2003/05/12
500126	10001	2003/05/13
500127	10004	2003/05/14

```
SELECT suppliers.supplier_id, suppliers.supplier_name, orders.order_date
FROM suppliers
FULL OUTER JOIN orders
ON suppliers.supplier_id = orders.supplier_id;
```