Table Setup

```
CREATE TABLE employees_v2 (
    EmpID INT PRIMARY KEY,
    EmpName VARCHAR(100),
    Department VARCHAR(50),
    City VARCHAR(50),
    Salary INT,
    HireDate DATE
);

INSERT INTO employees_v2 (EmpID, EmpName, Department, City, Salary, HireDate)
VALUES
(101, 'Rahul Mehta', 'Sales', 'Delhi', 55000, '2020-04-12'),
(102, 'Priya Sharma', 'HR', 'Mumbai', 62000, '2019-09-25'),
(103, 'Aman Singh', 'IT', 'Bengaluru', 72000, '2021-03-10'),
(104, 'Neha Patel', 'Sales', 'Delhi', 48000, '2022-01-14'),
(105, 'Karan Joshi', 'Marketing', 'Pune', 45000, '2018-07-22'),
(106, 'Divya Nair', 'IT', 'Chennai', 81000, '2019-12-11'),
(107, 'Raj Kumar', 'HR', 'Delhi', 60000, '2020-05-28'),
(108, 'Simran Kaur', 'Finance', 'Mumbai', 58000, '2021-08-03'),
(109, 'Arjun Reddy', 'IT', 'Hyderabad', 70000, '2022-02-18'),
(110, 'Anjali Das', 'Sales', 'Kolkata', 51000, '2023-01-15');
```

1. Show employees working in either the ‘IT’ or ‘HR’ departments.

➔ Snippet to run:

```
SELECT * FROM employees_v2

WHERE Department IN ('IT', 'HR');
```

EmpID	EmpName	Department	City	Salary	HireDate
102	Priya Sharma	HR	Mumbai	62000	2019-09-25
103	Aman Singh	IT	Bengaluru	72000	2021-03-10
106	Divya Nair	IT	Chennai	81000	2019-12-11
107	Raj Kumar	HR	Delhi	60000	2020-05-28
109	Arjun Reddy	IT	Hyderabad	70000	2022-02-18

- **WHERE Department IN ('IT', 'HR')**: The `IN` operator is a shorthand for multiple `OR` conditions. It checks if the value in the `Department` column matches any value inside the parentheses.
- **Efficiency**: Using `IN` is much cleaner than writing `WHERE Department = 'IT' OR Department = 'HR'.`

2. Retrieve employees whose department is in ‘Sales’, ‘IT’, or ‘Finance’.

➔ Snippet to run:

```
SELECT * FROM employees_v2

WHERE Department IN ('Sales', 'IT', 'Finance');
```

EmpID	EmpName	Department	City	Salary	HireDate
101	Rahul Mehta	Sales	Delhi	55000	2020-04-12
103	Aman Singh	IT	Bengaluru	72000	2021-03-10
104	Neha Patel	Sales	Delhi	48000	2022-01-14
106	Divya Nair	IT	Chennai	81000	2019-12-11
108	Simran Kaur	Finance	Mumbai	58000	2021-08-03

- **Multiple Values**: The `IN` operator allows you to specify a list of values, and MySQL will return any row where the `Department` matches any item in that list.
- **Logic**: This is equivalent to writing `Department = 'Sales' OR Department = 'IT' OR Department = 'Finance'.`

3. Display employees whose salary is between ₹50,000 and ₹70,000.

➔ Snippet to run:

```
SELECT * FROM employees_v2

WHERE Salary BETWEEN 50000 AND 70000;
```

EmpID	EmpName	Department	City	Salary	HireDate
101	Rahul Mehta	Sales	Delhi	55000	2020-04-12
102	Priya Sharma	HR	Mumbai	62000	2019-09-25
107	Raj Kumar	HR	Delhi	60000	2020-05-28
108	Simran Kaur	Finance	Mumbai	58000	2021-08-03
109	Arjun Reddy	IT	Hyderabad	70000	2022-02-18
110	Anjali Das	Sales	Kolkata	51000	2023-01-15

- **BETWEEN ... AND ...**: This operator filters the result set to include values within a specific range.
- **Inclusive Nature**: In SQL, the `BETWEEN` operator is **inclusive**, meaning it will include employees who earn exactly ₹50,000 or ₹70,000.
- **Applicability**: While used here for numbers (Salary), this operator also works effectively for dates and text.

4. List employees whose names start with the letter ‘A’.

➔ Snippet to run:

```
SELECT * FROM employees_v2

WHERE EmpName LIKE 'A%';
```

EmpID	EmpName	Department	City	Salary	HireDate
103	Aman Singh	IT	Bengaluru	72000	2021-03-10
109	Arjun Reddy	IT	Hyderabad	70000	2022-02-18
110	Anjali Das	Sales	Kolkata	51000	2023-01-15
NULL	NULL	NULL	NULL	NULL	NULL

- **LIKE 'A%'**: The `LIKE` operator is used to search for a specified pattern in a column.
- **The % Wildcard**: Placing the `%` after 'A' tells MySQL to find any name that begins with "A" followed by any sequence of characters.
- **Case Sensitivity**: In most MySQL configurations, `LIKE` is case-insensitive, but it is standard practice to match the casing of your data.

5. Find employees whose names contain the substring ‘an’.

➔ Snippet to run:

```
SELECT * FROM employees_v2

WHERE EmpName LIKE '%an%';
```

EmpID	EmpName	Department	City	Salary	HireDate
103	Aman Singh	IT	Bengaluru	72000	2021-03-10
105	Karan Joshi	Marketing	Pune	45000	2018-07-22
108	Simran Kaur	Finance	Mumbai	58000	2021-08-03
110	Anjali Das	Sales	Kolkata	51000	2023-01-15

- **LIKE '%an%'**: The % wildcard placed at the beginning and end of the string tells MySQL to match any sequence of characters, followed by "an", followed by any other sequence.
- **Flexibility**: This pattern will find names where "an" appears at the start (e.g., **An**jali), the middle (e.g., K**an**ti), or the end (e.g., Rom**an**).

6. Show employees who are from ‘Delhi’ or ‘Mumbai’ and earn more than ₹55,000.

➔ Snippet to run:

```
SELECT * FROM employees_v2

WHERE (City = 'Delhi' OR City = 'Mumbai')

AND Salary > 55000;
```

	EmpID	EmpName	Department	City	Salary	HireDate
▶	102	Priya Sharma	HR	Mumbai	62000	2019-09-25
	107	Raj Kumar	HR	Delhi	60000	2020-05-28
	108	Simran Kaur	Finance	Mumbai	58000	2021-08-03
*	NULL	NULL	NULL	NULL	NULL	NULL

(City = 'Delhi' OR City = 'Mumbai'): The parentheses group these two cities together, telling MySQL that the employee must belong to at least one of these locations.

AND Salary > 55000: This second rule is mandatory; even if they live in the correct city, they will be excluded if their salary is 55,000 or less.

Logic Precedence: Without parentheses, SQL might evaluate `City = 'Mumbai' AND Salary > 55000` first, which would change your results.

7. Display all employees except those from the 'HR' department.

➔ Snippet to run:

```
SELECT * FROM employees_v2

WHERE Department != 'HR';
```

EmpID	EmpName	Department	City	Salary	HireDate
101	Rahul Mehta	Sales	Delhi	55000	2020-04-12
103	Aman Singh	IT	Bengaluru	72000	2021-03-10
104	Neha Patel	Sales	Delhi	48000	2022-01-14
105	Karan Joshi	Marketing	Pune	45000	2018-07-22
106	Divya Nair	IT	Chennai	81000	2019-12-11
108	Simran Kaur	Finance	Mumbai	58000	2021-08-03
109	Arjun Reddy	IT	Hyderabad	70000	2022-02-18
110	Anjali Das	Sales	Kolkata	51000	2023-01-15
NULL	NULL	NULL	NULL	NULL	NULL

- **The `!=` Operator**: This operator filters the results to only include rows where the `Department` value is anything *other* than "HR".
- **Alternative Syntax**: You can also use `<>`, which is the standard SQL operator for "not equal to".
- **Impact on Results**: This query will return records for employees in Sales, IT, Marketing, and Finance, while completely hiding those in HR.

8. Get all employees hired between 2019 and 2022, ordered by HireDate (oldest first)

➔ Snippet to run:

```
SELECT * FROM employees_v2

WHERE HireDate BETWEEN '2019-01-01' AND '2022-12-31'

ORDER BY HireDate ASC;
```

EmpID	EmpName	Department	City	Salary	HireDate
102	Priya Sharma	HR	Mumbai	62000	2019-09-25
106	Divya Nair	IT	Chennai	81000	2019-12-11
101	Rahul Mehta	Sales	Delhi	55000	2020-04-12
107	Raj Kumar	HR	Delhi	60000	2020-05-28
103	Aman Singh	IT	Bengaluru	72000	2021-03-10
108	Simran Kaur	Finance	Mumbai	58000	2021-08-03
104	Neha Patel	Sales	Delhi	48000	2022-01-14
109	Arjun Reddy	IT	Hyderabad	70000	2022-02-18
NULL	NULL	NULL	NULL	NULL	NULL

- **WHERE HireDate BETWEEN...**: This selects any employee whose HireDate falls within the four-year window specified.
- **'2019-01-01' AND '2022-12-31'**: We use the standard SQL date format (YYYY-MM-DD) to ensure the filter covers everything from New Year's Day 2019 to New Year's Eve 2022.
- **ORDER BY HireDate ASC**: This sorts the results starting with the earliest date (Oldest First).