**Database Interview Questions**

1: What is the difference between primary key and unique key?

Write a DDL statement using primary key and unique key.

Ans: A primary key and a unique key both enforce uniqueness of the values in a column or set of columns, but they are used in different ways:

Primary Key:

A primary key is a unique identifier for a record in a table.

It must contain unique values, and it cannot contain null values.

A table can have only one primary key.

Unique Key:

A unique key is similar to a primary key, but a table can have multiple unique keys.

Like the primary key, a unique key must also contain unique values, but it can contain null values.

Here is a sample DDL statement that creates a table with both a primary key and a unique key:

CREATE TABLE student (

student\_id INT NOT NULL,

student\_name VARCHAR (100) NOT NULL,

student\_email VARCHAR (100) NOT NULL,

PRIMARY KEY (student\_id),

UNIQUE (student\_email)

);

2: How to read TOP 2 and LAST 2 records from a table using a SQL query?

|  |  |  |
| --- | --- | --- |
| Emp Id | Emp Name | Age |
| 1 | John | 26 |
| 2 | Edward | 30 |
| 3 | Thomas | 40 |
| 4 | Percy | 30 |
| 3 | Thomas | 40 |

**Ans**: The following SQL query can be used to read the top 2 records from a table:

SELECT \*

FROM table\_name

LIMIT 2;

And the following query can be used to read the last 2 records from a table:

SELECT \*

FROM table\_name

ORDER BY Emp Id DESC

LIMIT 2;

Note: In the above queries, replace "table\_name" with the actual name of your table. Also, make sure to adjust the **ORDER BY** clause in the second query based on the actual column name that you want to sort the records by (in this case, "Emp Id").

3. Write Sql query to fetch duplicate rows in the below table. Also to remove duplicate rows from the table.

|  |  |  |
| --- | --- | --- |
| Emp Id | Emp Name | Age |
| 1 | John | 26 |
| 2 | Edward | 30 |
| 3 | Thomas | 40 |
| 4 | Percy | 30 |
| 5 | Thomas | 40 |
|  |  |  |

4. What are the different types of JOIN?

Ans: There are several types of JOINs in SQL, including:

1. INNER JOIN: Returns only the rows that have matching values in both tables.
2. LEFT JOIN (or LEFT OUTER JOIN): Returns all the rows from the left table
3. (table1), and the matching rows from the right table (table2). The result set will contain NULL values for non-matching rows from the right side table.
4. RIGHT JOIN (or RIGHT OUTER JOIN): Returns all the rows from the right table (table2), and the matching rows from the left table (table1). The result set will contain NULL values for non-matching rows from the left side table.
5. FULL OUTER JOIN: Returns all the rows from both tables, including the matching and non-matching rows. If a row exists in one table and not in the other, the result set will contain NULL values for the non-matching columns.
6. CROSS JOIN: Returns the Cartesian product of two tables, which is a result set that contains every possible combination of rows from both tables.

It's important to choose the right type of JOIN based on the requirements of the query, as it can significantly impact the performance and accuracy of the result set.

1. How would you write a query to JOIN these two tables so that all the columns form employee table must be present in the output?

We have two tables:

Employees - This table contains each employee’s ID, name, and department ID.

|  |  |  |
| --- | --- | --- |
| Id | Employee\_name | Department\_id |
| 1 | Toby | 4 |
| 2 | Thomas | 1 |
| 3 | Edward | 5 |
| 4 | Percy | 3 |
| 5 | james | Null |

departments - This table contains each department’s ID and name.

|  |  |
| --- | --- |
| Department-id | Department-name |
| 1 | Sales |
| 2 | Engineering |
| 3 | Human Resources |
| 4 | Customer Services |
| 5 | R&D |

Java Interview Questions

**Guess The output :**

Question

1.

public class Leaf {

int i = 0;

Leaf increment() {

i++;

return this;

}

void print() {

System.*out*.println("i = " + i);

}

public static void main(String[] args) {

Leaf x = new Leaf();

x.increment().increment().increment().print();

}

}

Ans2. The output of the above Java program would be:

i = 3

Here's how it works:

* A **Leaf** object is created and assigned to **x**.
* The **increment** method is called three times on the **x** object, incrementing the value of **i** by 1 each time.
* Finally, the **print** method is called on the **x** object, which prints the value of **i** to the console.

So, the output would be "i = 3".

2. public class Dog {

public Dog() {

System.*out*.println("Tail");

}

static {

System.*out*.println("Paws");

}

public static void main(String[] args) {

Dog d=new Dog();

}

}

Ans 2. The output of the above Java program would be:

Paws Tail

Here's how it works:

* The **Dog** class has a static block that prints "Paws" when the class is first loaded by the JVM.
* The **Dog** class also has a constructor that prints "Tail".
* In the **main** method, a **Dog** object is created and assigned to **d**, which triggers the constructor to be called and prints "Tail".

So, the output would be "Paws\nTail".

3. public class Base {

static int *i*=10;

static{

*m1*();

System.*out*.println("First static block");

}

public static void main(String[] args) {

*m1*();

System.*out*.println("main method");

}

public static void m1() {

System.*out*.println("j = "+*j*);

}

static {

System.*out*.println("second static block");

}

static int *j*=20;

}

Ans 3

The output of the above Java code will be:

j = 0 First static block j = 20 second static block main method

4. public class Demo {

public static void main(String[] args) {

System.*out*.println(Left.*x*);

System.*out*.println(Right);

System.*out*.println(x);

}

}

interface Left {

int *x*=777;

}

interface Right {

int *x*=888;

}

Ans 4. The code above is a Java program that declares a class "Demo" with a "main" method, and two interfaces "Left" and "Right". The "main" method contains three calls to "System.out.println", which print the values of three variables named "x".

The two interfaces, "Left" and "Right", both declare an integer constant "x" with the value of 777 and 888, respectively.

However, there is no declaration of the variable "x" in the "Demo" class. So, the third call to "System.out.println(x)" would result in a compile-time error, with a message similar to "error: cannot find symbol x".

If you were to fix the error by declaring "x" in the "Demo" class, the program would print

777

888

5.

class Grain {

public String toString() { return "Grain"; }

}

class Wheat extends Grain {

public String toString() { return "Wheat"; }

}

class Mill {

Grain process() { return new Grain(); }

}

class WheatMill extends Mill {

Wheat process() { return new Wheat(); }

}

public class Sample {

public static void main(String[] args) {

Mill m = new Mill();

Grain g = m.process();

System.*out*.println(g);

m = new WheatMill();

g = m.process();

System.*out*.println(g);

}

}

Ans 5. When the program is run, the following will be the output:

Grain Wheat

**Write the logic**

Questions

1 .Given an array of integers nums sorted

 in non-decreasing order, find the starting and ending position of a given target value.

If target not found in the array, return[-1,-1]

**Input** nums = [5,7,7,8,8,10], target = 8

**Output** [3,4]

Here is the logic to solve this problem:

1. Initialize two variables **start** and **end** to store the starting and ending position of the target value.
2. Use binary search to find the first occurrence of the target value in the array. Set **start** to the index where the target value is found. If the target value is not found, return [-1, -1].
3. Use binary search to find the last occurrence of the target value in the array. Set **end** to the index where the target value is found. If the target value is not found, return [-1, -1].
4. Return the **[start, end]** as the result.

def searchRange(nums, target):

def binarySearch(nums, target, left):

right = len(nums) - 1

while left <= right:

mid = (left + right) // 2

if nums[mid] == target:

return mid

elif nums[mid] < target:

left = mid + 1

else:

right = mid - 1

return -1

start = binarySearch(nums, target, 0)

if start == -1:

return [-1, -1]

end = binarySearch(nums, target, start + 1)

if end == -1:

end = start

else:

end -= 1

return [start, end]

Example

nums = [5, 7, 7, 8, 8, 10]

target = 8

print(searchRange(nums, target)) # Output: [3, 4]

2. Minimum number of platform required for the train

Problem Statement:We are given two arrays that represent the arrival and

departure times of trains that stop at the platform. We need to find

the minimum number of platforms needed at the railway station so that no train has to wait.

Example

**Input:** N=6,

arr[] = {9:00, 9:45, 9:55, 11:00, 15:00, 18:00}

dep[] = {9:20, 12:00, 11:30, 11:50, 19:00, 20:00}