Javarevited-2016

**How to declare and initialize a List (ArrayList and LinkedList) with values in Java**  
**It's important to remember that List returned by Arrays.asList() can not be used as regular List for further adding or removing elements. It's kind of fixed length Lists which doesn't support addition and removal of elements.** If you want to convert that fixed length List into a proper ArrayList, LinkedList or Vector any other Collection class you can always use the copy constructor provided by Collection interface, which allows you to pass a collection while creating ArrayList or LinkedList and all elements from source collection will be copied over to the new List. This will be the shallow copy so beware, any change made on an object will reflect in both the list.  
  
**4 Ways to find Nth highest salary in SQL - Oracle, MSSQL and MySQL**  
  
Use below query to create table and build schema:  
-- creating Employee table in Oracle

CREATE TABLE Employee (name varchar(10), salary int);

-- inserting sample data into Employee table

INSERT INTO Employee VALUES ('Rick', 3000);

INSERT INTO Employee VALUES ('John', 4000);

INSERT INTO Employee VALUES ('Shane', 3000);

INSERT INTO Employee VALUES ('Peter', 5000);

INSERT INTO Employee VALUES ('Jackob', 7000);  
**SQL Query:**

SELECT name, salary

FROM #Employee e1

WHERE N-1 = (SELECT COUNT(DISTINCT salary) FROM #Employee e2

WHERE e2.salary > e1.salary)  
for the 2nd maximum you can replace N with 2, and for 3rd maximum replace N with 3, here is the output:  
  
**2nd highest salary:**

**SELECT name, salary**

**FROM #Employee e1**

**WHERE N-1 = (SELECT COUNT(DISTINCT salary) FROM #Employee e2**

**WHERE e2.salary > e1.salary)SELECT name, salary**

**FROM #Employee e1**

WHERE 2-1 = (SELECT COUNT(DISTINCT salary) FROM #Employee e2

WHERE e2.salary > e1.salary)

Result:

name salary

Peter 5000

**3rd highest salary:**

SELECT name, salary

FROM #Employee e1

WHERE 3-1 = (SELECT COUNT(DISTINCT salary) FROM #Employee e2

WHERE e2.salary > e1.salary)

Result:

name salary

John 4000

**Explanation :**  
The **distinct**keyword is there to deal with duplicate salaries in the table. In order to find the Nth highest salary, we are only considering unique salaries. Highest salary means no salary is higher than it, Second highest means only one salary is higher than it, 3rd highest means two salaries are higher than it, similarly Nth highest salary means N-1 salaries are higher than it.  
  
**Pros :**  
1) The generic solution works in all database including Oracle, MySQL, SQL SERVER and PostgreSQL.  
  
**Cons :**  
1) Slow, because the inner query will run for every row processed by the outer query.  
**Nth maximum salary in MySQL using LIMIT keyword**

Similar to TOP, MySQL also supports a LIMIT keyword, which provides pagination capability. You can find the nth highest salary in MySQL without using subquery as shown below:

SELECT salary FROM Employee ORDER BY salary DESC LIMIT N-1, 1  
2nd highest salary in MySQL without subquery:  
SELECT salary FROM Employee ORDER BY salary DESC LIMIT 1,1

salary

5000  
3rd highest salary in MySQL using LIMIT clause:  
SELECT salary FROM Employee ORDER BY salary DESC LIMIT 2,1

salary

4000

Nth highest salary in MySQL using LIMIT clause:  
SELECT salary FROM Employee ORDER BY Salary DESC LIMIT n-1,1  
**Explanation :**  
The benefit of this approach is that it's faster than correlated query approach but its vendor dependent. This solution will only work in MySQL database.  
**Nth highest salary in Oracle using ROW\_NUMBER() function**

SELECT \* FROM (

SELECT e.\*,

ROW\_NUMBER() OVER (ORDER BY salary DESC) rn

FROM Employee e

)

WHERE rn = N; /\*N is the nth highest salary\*/  
Here is the [2nd highest salary in Oracle using ROW\_NUMBER()](http://javarevisited.blogspot.com/2015/11/2nd-highest-salary-in-oracle-using-rownumber-rank-example.html) window function:  
SELECT \* FROM (

SELECT e.\*,

ROW\_NUMBER() OVER (ORDER BY salary DESC) rn

FROM Employee e

)

WHERE rn = 2;

Output

NAME SALARY RN

Peter 5000 2

and here is 3rd highest salary in Oracle:  
SELECT \* FROM (

SELECT e.\*,

ROW\_NUMBER() OVER (ORDER BY salary DESC) rn

FROM Employee e

)

WHERE rn = 3;   
By the above code has a problem. It is not handling duplicate salaries properly. For example, in our table we have two employees with salary 3000, that's our 4th highest salary, but above code will print the same salary, albeit different employee for both 4th and 5th maximum as shown below:

SELECT \* FROM ( SELECT e.\*, ROW\_NUMBER() OVER (ORDER BY salary DESC) rn FROM Employee e

) WHERE rn = 5;

Result:

NAME SALARY RN

Shane 3000 5  
In oracle, you can also use SQL statements to build schema and run sample SQL.  
  
You can also do the same thing by using **RANK()** window function in Oracle, but that's for another day. This is more than enough to answer the SQL interview question, the print nth highest salary of an employee in the Oracle.  
  
**2 ways to Split String with Dot (.) in Java using Regular Expression**  
  
First try  
Most of the Java programmer first try the following approach when they need to split the String on dot character:

String textfile = "ReadMe.txt";

String filename = textfile.split(".")[0];

String extension = textfile.split(".")[1];

**This will not work because dot (.) is a special character in Java regular expression** to match any single character. Above code will throw [java.lang.ArrayIndexOutOfBoundsException: 0](http://javarevisited.blogspot.com/2016/02/solving-javalangarrayindexoutofboundsexception-0-1-2-in-java.html" \t "_blank) because split() will return an empty array, as shown below:  
  
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0 at StringSplitWithRegEx.main(StringSplitWithRegEx.java:9)  
  
The problem with this code is that **"."** is a *metacharacter* if you want to use it literally *you need to escape it by using backslash* e.g. **\\.** , though you should remember that to escape dot you just need one backslash i.e **\.**, but in Java since \ backslash also need escaping [you need two backslashes](http://javarevisited.blogspot.com/2014/02/stringtokenizer-example-in-java-multiple-delimiters.html) or \\, as shown below:

String textfile = "ReadMe.txt";

String filename = textfile.split("\\.")[0];

String extension = textfile.split("\\.")[1];

Alternatively, you can also use the [.] regular expression to split the String by dots in Java, as shown below:  
String extension = "minecraft.exe".split("[.]")[1];  
The reason [.] work because the dot is **inside character class i.e. double brackets []**. Only characters **]^-\** have special meaning inside character classes in Java and dot is not one of them, which means you can use it literally inside [character class](http://javarevisited.blogspot.com/2012/10/regular-expression-example-in-java-to-check-String-number.html) or [ ].  
Similarly another example is given below.  
String pipeDelimited **=** "IBM|Intel|HP|Cisco";

String[] companies **=** pipeDelimited.**split**("|");

This will not yield the result you want i.e. an array of IBM, Intel, HP and Cisco, instead it will printt [, I, B, M, |, I, n, t, e, l, |, H, P, |, C, i, s, c, o], because | is interepreted as [logical operator](http://javarevisited.blogspot.com/2015/01/difference-between-bitwsie-and-logical.html) OR and Java regex engine splitted  
  
In order to solve this problem, you need to escape the pipe character when you pass to split() function as shown below:

**pipeDelimited.split("\\|"); // return [IBM, Intel, HP, Cisco]**