<https://www.softwaretestinghelp.com/how-to-write-test-cases-for-a-given-scenario/>

### Question #1)

This comes to us from “Bhawana Kumara” in response to this post: [*7 Types of Software Errors That Every Tester Should Know*](https://www.softwaretestinghelp.com/types-of-software-errors/)

***“At one of the interview I got this question, Write as many as test cases for – If you are a new customer and you want to open a credit card account then there are three conditions first you will get a 15% discount on all your purchases today, second if you are an existing customer and you hold a loyalty card, you get a 10% discount and third if you have a coupon, you can get 20% off today (but it can’t be used with the ‘new customer’ discount). Discount amounts are added, if applicable.***

***Can somebody please help me with it.”***

Sure. Happy to help!

**Answer:** This question is a classic case of different processing happening for different kinds of input. The input here is the type of the customer. The processing is the amount of discount that they can avail. Depending on the type of the input if the output differs, a model that can be successfully used to test is “Decision Table Testing”.

Let’s see how you can do this.

**Step 1: Partition your input into categories**

To create a decision table, you will have to partition your input into categories.

There are 6 categories of users in this situation:

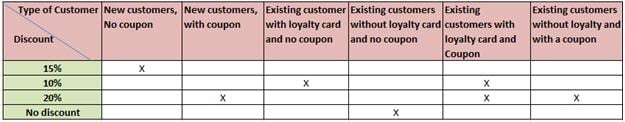
* New customers with coupon
* New customers without coupon
* Existing customers with loyalty card and no coupon
* Existing customers without loyalty card and no coupon
* Existing customers with loyalty card and Coupon
* Existing customers without loyalty and with a coupon

More partitions can be made but from the problem definition itself, it is not clear if the new customers can have a loyalty card or not. So let’s not assume it. Besides, this is just to show you how to arrive at a solution.

**Step 2: Construct your decision table.**

There are many ways to do this. I would use all the input categories as columns and discounts as rows. You end up with the following table:

**(Click on the below image for enlarged view)**

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2016/05/decision-table.jpg)

**Step 3: Pick a user from each input category and test**

Now from each category, you can pick one value and test to see if the correct amount of discount is applied.  
So now, you will need at least 6 customers or 6 test cases to test the case completely.

I’m sure at this point you are thinking, “This is all good. **But how can I answer the question in an interview instantly when I do not have the time to work out the detailed solution like you did?”**

This is why it is important to talk your thoughts in an interview.

As soon as you hear the question, you can say: I think a decision table will help solve this problem. If the interviewer wants you to elaborate, you can ask for a notepad and a pen and work it out. Be sure to explain your solution as you go.

Also, keep in mind that, it is not important to get the solution a 100% right. So you might miss an input category or two, because of the pressure in an interview or in haste and that is OK. The interviewer will be appreciative of your strategy and clarity of thought.

### Question #2)

Dhrumil posted the question: “***What is 80:20 rule? Please explain through Example***.” in response to [*All In One Guide On Defect Density – Its Importance and How to Calculate It*](https://www.softwaretestinghelp.com/defect-density/).

**Answer:** 80/20 rule is also called the Pareto principle. You can check out a basic definition here at [Pareto principle](https://en.wikipedia.org/wiki/Pareto_principle)

It says 80% of the results are due to 20% of the causes. It can be applied to many things and in the context of an IT QA project, the following may be observed.

* 80% of your productivity is due to 20% of the activities that you do
* 80% of progress is contributed by 20% of your team
* 80% of the application can be tested by 20% of the test cases
* 80% of the malfunctions can be addressed by fixing 20% of your defects

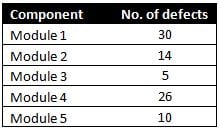
And, so on.

Therefore, following this rule, we will have to identify what that 20% of the causes.

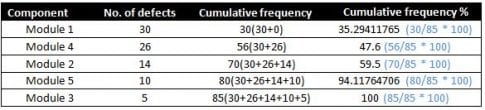
Pareto analysis is simply a technique that helps you optimize your efforts. Instead of distributing your focus and effort on all 100% causes, it tells us to look for that 20% of the causes which when addressed maximize your returns (80% of the problems).

* This is not an exact science and should not be taken at face value.
* Pareto analysis finds its usage in many industries and not just software.
* To know exactly what that 20% of the causes are, you could draw up a **Pareto chart**. It is a simple combination of both bar and line charts that plots the causes on the X axis and problems on the Y axis. The cumulative frequency line point of 80% is dropped onto the X axis. All the causes that lie in between 0 and the line are the 20% causes.

For example: if there are 5 modules in an application that have defects to be fixed and this is the distribution of them:

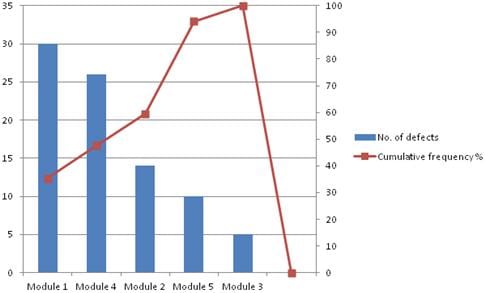


You will now reorder this table in a descending order of defect count and calculate the cumulative frequency percentage.

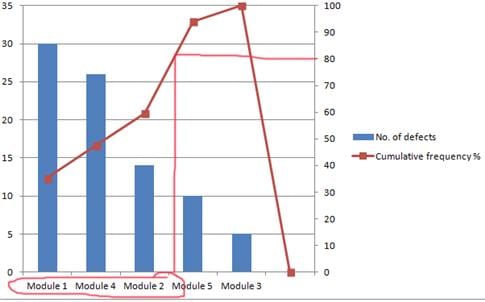


When we draw a Pareto Chart for the above tabular data, this is what you will end up with:

(For the exact steps as to how to generate a Pareto Chart in Excel follow this link: [Pareto Chart](http://www.excel-easy.com/examples/pareto-chart.html))

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2016/05/cumulative-frequency.jpg)

Now, to know what your 20% of the modules are to fix in order to target optimum defect fixing, draw a line at the 80% mark on your cumulative frequency percentage axis, like below:

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2016/05/cumulative-frequency-1.jpg)

Hence, the 20% modules you should focus on are Module 1, 4 and 2.

This is, of course, a contrived example, but when the number of causes is in a few 10’s this method can do wonders in terms of giving us the clarity on what to pick.