Equivalence Paritioning:

Divide the input domain: Identify the different sets of inputs based on the software's requirements. These sets are divided into partitions of equivalent data, which are often categorized as either valid or invalid.

•**Create test cases:** Choose one representative value from each of these partitions to create a single test case. For example, if a system accepts numbers between 1 and 100, the partitions might be:

•**Valid:** 1-100

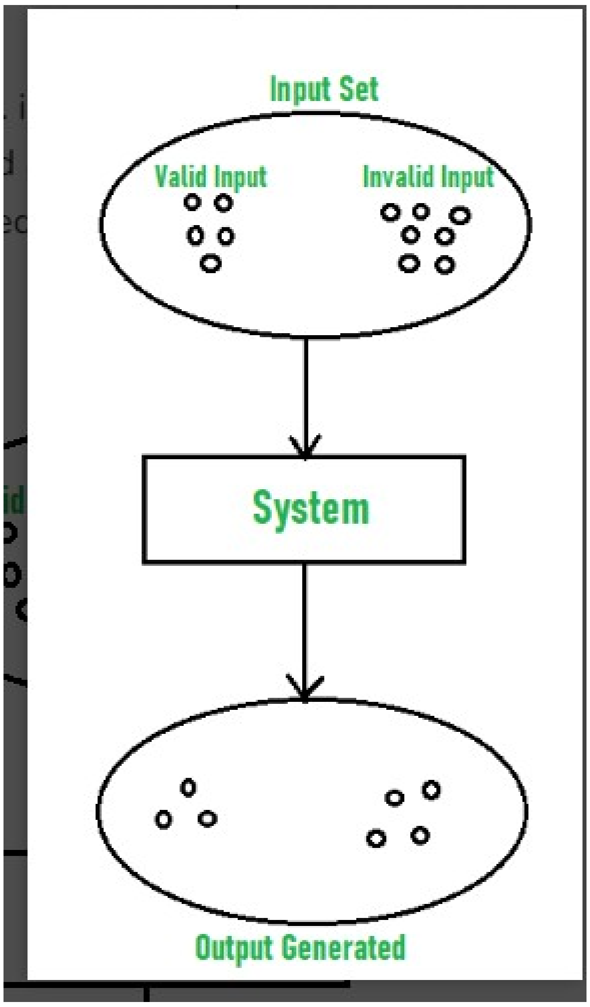
•**Invalid:** 0 or less

•**Invalid:** Greater than 100

Example: Age validation

A common example is testing a form that requires a user's age to be between 20 and 60.

- **Valid partition:** Ages 20-60. A test case could be a user aged 45.
- **Invalid partition:** Ages 0-19. A test case could be a user aged 15.
- **Invalid partition:** Ages 61 and above. A test case could be a user aged 70.



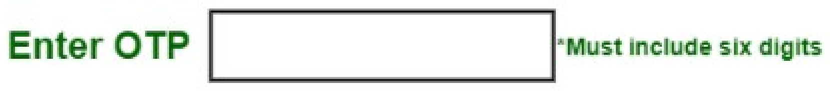
Guidelines for Equivalence Partitioning :

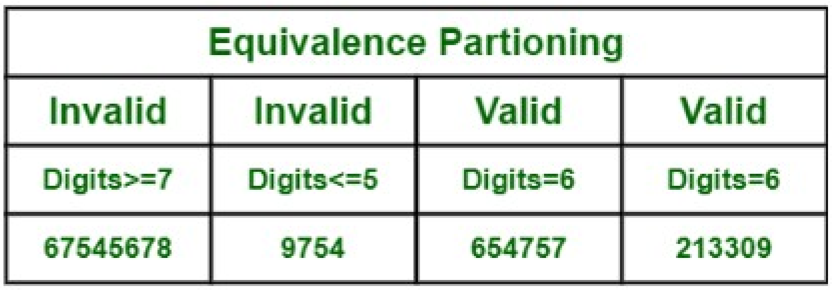
•If the range condition is given as an input, then one valid and two invalid equivalence classes are defined.

•If a specific value is given as input, then one valid and two invalid equivalence classes are defined.

•If a member of set is given as an input, then one valid and one invalid equivalence class is defined.

•If Boolean no. is given as an input condition, then one valid and one invalid equivalence class is defined.





Equivalence Partioning			
Invalid	Invalid	Valid	Valid
Digits>=7	Digits<=5	Digits=6	Digits=6
67545678	9754	654757	213309

Boundary Value Analysis:

Boundary value analysis is a software testing technique that focuses on testing the limits of input ranges, such as the minimum and maximum values, because errors are often found at these boundaries.

<u>**Working:**</u>

Identify boundaries: For a given input range (e.g., age from 18 to 60), the minimum is 18 and the maximum is 60.

•**Test values**: Create test cases using values around the boundaries:

•**Minimum boundary**: Test the minimum value itself, one value below it, and one value above it (e.g., 17, 18, 19).

•**Maximum boundary**: Test the maximum value itself, one value below it, and one value above it (e.g., 59, 60, 61).

•**Test valid and invalid inputs**: Ensure the software handles both valid boundary conditions (like 18 and 60) and invalid boundary conditions (like 17 and 61) correctly.

•**Focus on edge cases**: By testing these extreme values, you can uncover errors that result from off-by-one mistakes, such as using ">" instead of "≥" in a conditional statement.

Example:

For an application that requires a password between 7 and 15 characters:

- **Valid values**: Test with 7, 8, 14, and 15 characters.
- **Invalid values**: Test with 6 characters and 16 characters to ensure the system rejects them

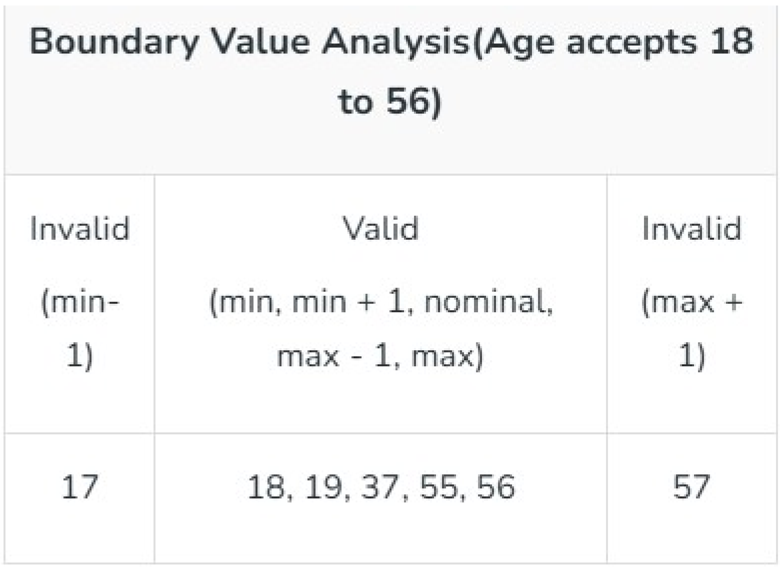
<u>Points to consider in BVA:</u>

A boundary value for a valid partition is a valid boundary value.

•A boundary value for an invalid partition is an invalid boundary value.

•For each variable we check-

- Minimum value.
- Just above the minimum.
- Nominal Value.
- Just below Max value.
- Max value.



Boundary Value Analysis(Age accepts 18 to 56)		
Invalid (min-1)	Valid (min, min + 1, nominal, max - 1, max)	Invalid (max + 1)
17	18, 19, 37, 55, 56	57

Valid Test cases: Valid test cases for the above can be any value entered greater than 17 and less than 57.

•Enter the value- 18.

•Enter the value- 19.

•Enter the value- 37.

•Enter the value- 55.

•Enter the value- 56.

Invalid Testcases: When any value less than 18 and greater than 56 is entered.
•Enter the value- 17.

•Enter the value- 57.

The idea and motivation behind BVA are that errors tend to occur near the extremes of the variables. The defect on the boundary value can be the result of countless possibilities.