SOFTWARE TESTING

Manual Testing Concepts:

Introduction to Software Testing

Software Testing is important as it uncovers a defect before it is delivered to the customer ensuring the quality of software. So that the defects or bugs can be identified in the early stages of development; as later the stage in which bug is identified, more is the cost to rectify it.

In this article we are going to discuss about the following:

- What is Software testing?
- Types of software testing?
- What is Manual testing?
- Types of manual testing?
- How to Perform Manual testing?
- Misconception about Manual Testing

What is Software testing?

Software testing is the process of evaluating a software system or its components with the intent of finding defects (errors or other issues) and verifying that the system meets specified requirements. Testing helps to ensure that the software is of high quality, performs as expected, and is fit for its intended purpose.

There are various types of software testing, including unit testing, integration testing, system testing, and acceptance testing. Each type of testing has a specific focus and purpose.

Unit testing involves testing individual units or components of a software system in isolation from the rest of the system. The goal of unit testing is to validate that each unit of the software is working as intended.

Integration testing involves testing the integration of different units or components of the software system. The goal of integration testing is to ensure that the different units or components of the software work together as intended.

System testing involves testing the complete software system in its intended environment, simulating real-world scenarios. The goal of system testing is to ensure that the software system is working as intended and meets the specified requirements.

Acceptance testing involves testing the software system to determine whether it is acceptable for delivery to the end user. The goal of acceptance testing is to determine whether the software system meets the requirements and is fit for its intended purpose.

Software testing is an important part of the software development process because it helps to identify defects and ensure that the software is of high quality. It is typically conducted by a team of software testers who are responsible for designing and executing test cases, analyzing the results, and reporting any defects that are found.

Effective software testing requires a thorough understanding of the software system and the requirements it is intended to meet. It also requires careful planning and the use of appropriate testing techniques and tools.

There are many challenges involved in software testing, including the need to test a large number of scenarios, the complexity of modern software systems, and the ever-evolving nature of software development. To address these challenges, software testers must be highly skilled and have a deep understanding of software development processes and methodologies.

- Software testing is conduct to verify the application is working correctly or not.
- Testing is conduct to find the defect in application .
- To validate the application is working as per customer requirement or not.

Types of software testing?

Manual testing: *Human performs the tests step by step, without test scripts.*
Automated testing: *Tests are executed automatically via test automation frameworks, along with other tools and software.*

What is Manual testing?

- Manual Testing is a process of finding out the defects, bugs in a software program.
- A tester performs the end-user role and verifies if all the features are working properly or not.
- The Tester performs the tests step by step manually.
- By Conducting the testing we can assess the quality of the application.

Types of manual testing?

- Unit Testing
- Integration Testing
- Black Box Testing
- White Box Testing
- System Testing
- Acceptance Testing

How to Perform Manual testing?

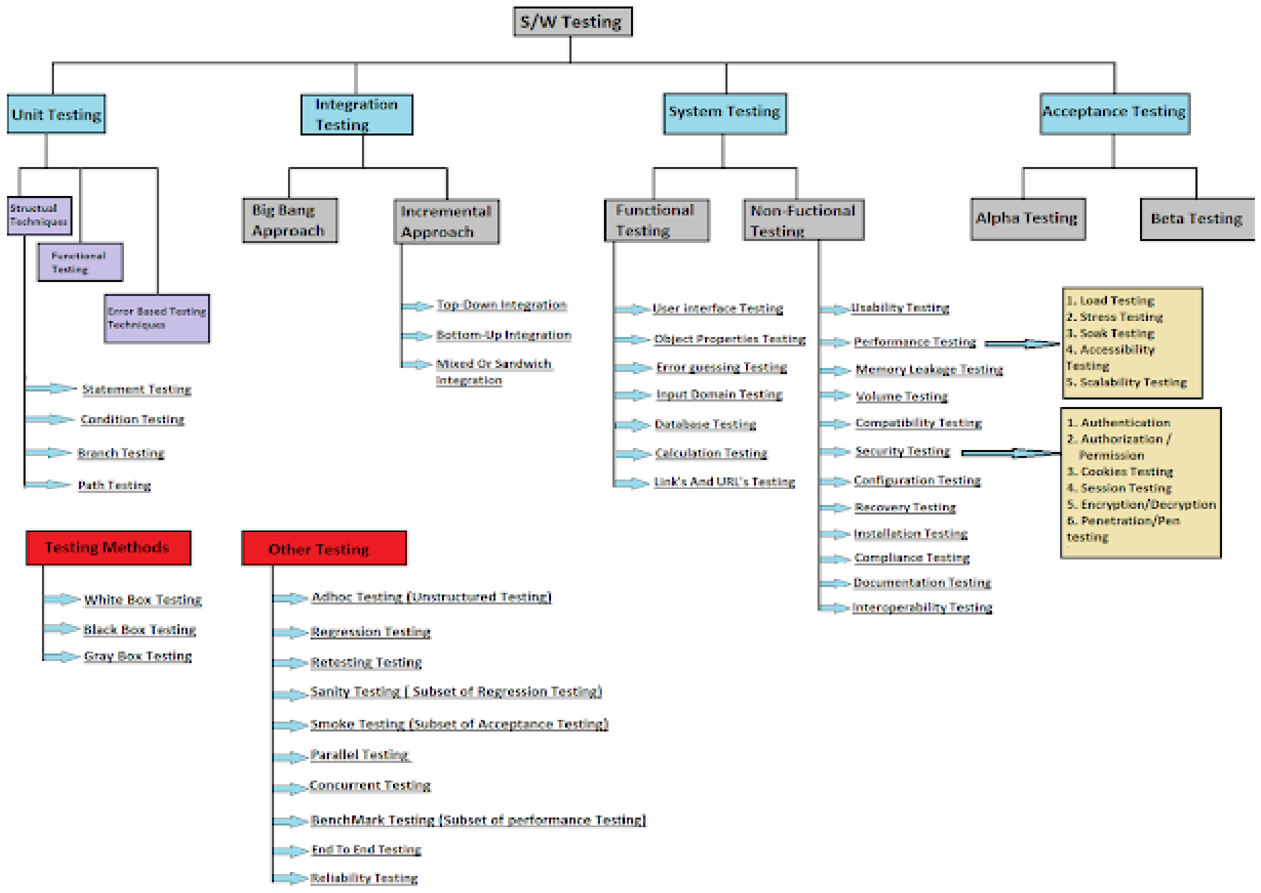
- Understand the requirements from the software requirement specification document.
- Identify the scenario and create a clear test plan.
- Design test cases that cover all the requirements defined in the document.
- Get test cases reviewed by the QA lead and update it based on reviews.
- Execute test cases and detect any bugs
- Report the defect to developer and track the status.
- Once bugs are fixed, again execute the failing test cases to verify they pass.

Misconception about Manual Testing.

- Myth #1: Anyone can do manual testing.
- Myth #2: Testing is just verifying the app from the UI
- Myth #3: Testing ensures 100% defect-free product.
- Myth #4: Automated testing is more powerful than manual testing.
- Myth #5: Manual testing is non-technical and easy.

Complete Software Testing Hierarchy | Software Testing Levels |Different Testing Types with Details

Complete Software Testing Hierarchy



Software Testing Hierarchy

SDLC (Software development life cycle)

Any Project Development has to follow the below phases :

- System Investigation
- Requirement analysis
- Design
- Coding
- Testing
- Release and maintenance

System Investigation

- It is the most important and necessary stage in SDLC
- Understand the existing business ,process.
- Gather all the data like what the customer wants to build, who will be the end user, what is the objective of the product.
- It is Business analyst’s and Project organizer’s Responsibility.

Requirement analysis

The next stage is to certainly represent and document the software requirements.
This is accomplished through BRD / FRD / SRS / FRS document which contains all the product requirements.
The Requirement collected are 2 type:

1. Functional requirements: Specify the behavior of the application.
2. Non-functional requirement: Specify the characteristic and features of the application like performance , compatibility, usability etc .

Design

- The next phase is about to bring down all the knowledge of requirements, analysis, and design of the software project.
- This phase is the product of the last two, like inputs from the customer and requirement gathering.

Coding

- In this phase the actual development begins, and the programming is built.
- It is the responsibility of the developers to write the coding based on design.
- Developers have to follow the coding guidelines described by their management and programming tools.

Testing

- After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage.
- During this stage, unit testing, integration testing, system testing, acceptance testing are done.

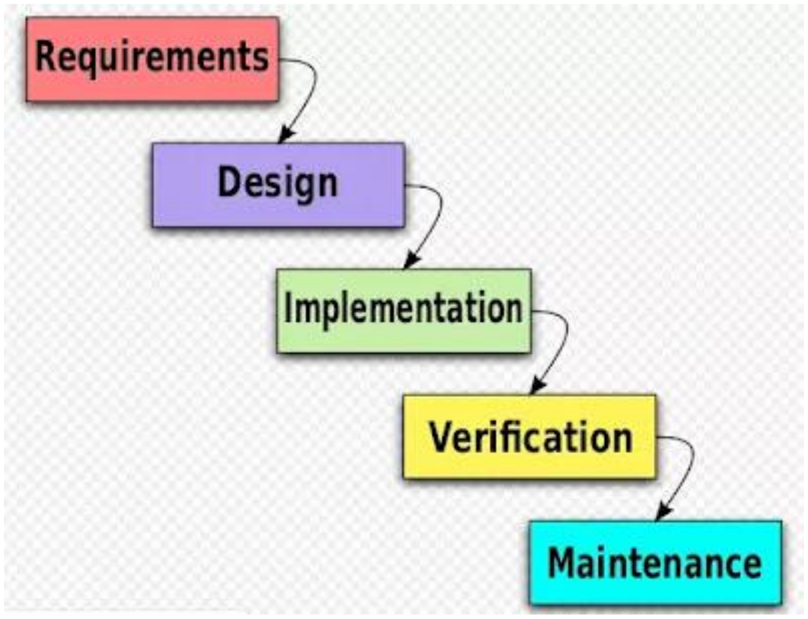
Release and Maintenance

- Once the software is certified, and no bugs or errors are stated, then it is deployed
- If any issue in the software or if any changes required it comes under the maintenance part
- The taking care of the developed product is known as maintenance
- The maintenance of the project can be taken place by supporting or post production team.

All Software Development Models

<u>Water fall model / Linear sequential model</u>

- The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete.
- Waterfall Model is a sequential model that divides software development into pre-defined phases. Each phase must be completed before the next phase can begin with no overlap between the phases.



Water Fall Model

When to go for waterfall model?

- Requirements are not changing frequently
- Application is not complicated and big
- Project is short
- Requirement is clear
- Environment is stable
- Technology and tools used are not dynamic and is stable

Advantages:

- This model is easy to understand easy to implement
- Is is suitable for small projects
- No changes in the middle of the project

Disadvantages:

- No feedback path: In this model It assumes that no error is ever committed by developers during any phases. Therefore, it does not incorporate any mechanism for error correction.
- Not a good model for complex and object-oriented projects.

- Poor model for long and ongoing projects.
- Cannot accommodate changing requirements.

Please refer the below video for more detail on waterfall model:

Iterative Model

- We create rough product or product piece in one iteration, then review it and improve it in next iteration and so on until it's finished.
- The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

Advantages:

- Testing and debugging during smaller iteration is easy.
- It is easily acceptable to ever-changing needs of the project.
- Limited time spent on documentation and extra time on designing.
- Risks are identified and resolved during iteration.

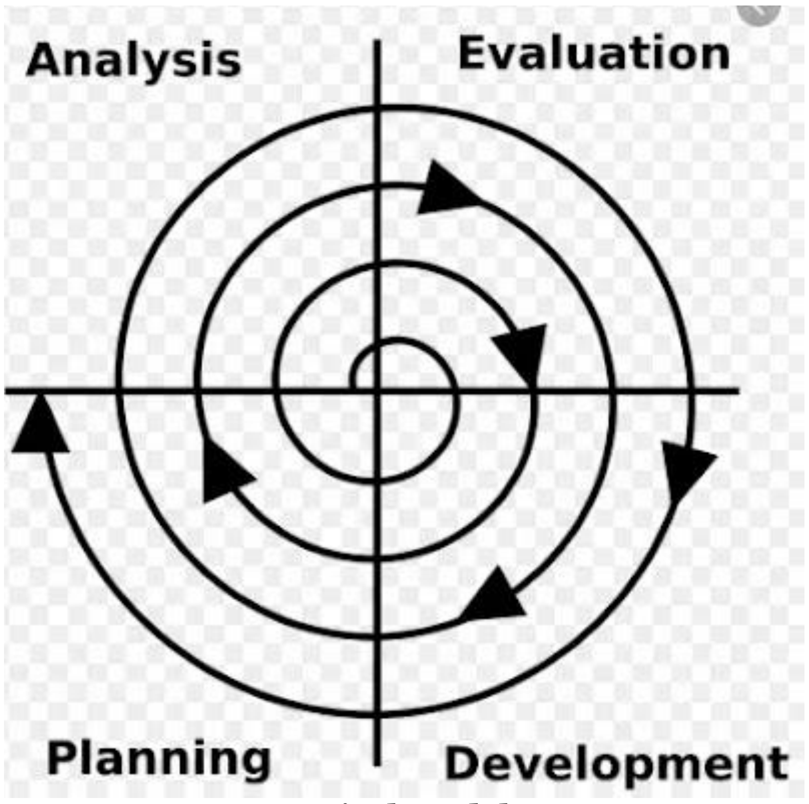
Disadvantages:

- It is not suitable for smaller projects.
- Design can be changed again and again because of imperfect requirements.
- Requirement changes can cause over budget.
- Project completion date not confirmed because of changing requirements.

Please refer the below video for more detail on Iterative model:

Spiral Model

The spiral model is a risk-driven software development process model. Based on the unique risk patterns of a given project, the spiral model guides a team to adopt elements of one or more process models, such as incremental, waterfall, or evolutionary prototyping.



Spiral Model

- It is incremental development approach. The project is developed phase by phase. The new requirements are integrated to the existing application.
- The requirements are prioritized based on business requirements.
- This is a risk-driven software development process model.
- It is suitable for large and complex projects.
- This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis.

When to use Spiral Model?

- When deliverance is required to be frequent.
- When Project is large.
- When requirements are unclear and complex.

Advantages:

- The requirement are prioritized based on the business requirement
- Is is suitable for large and complex projects

Disadvantages:

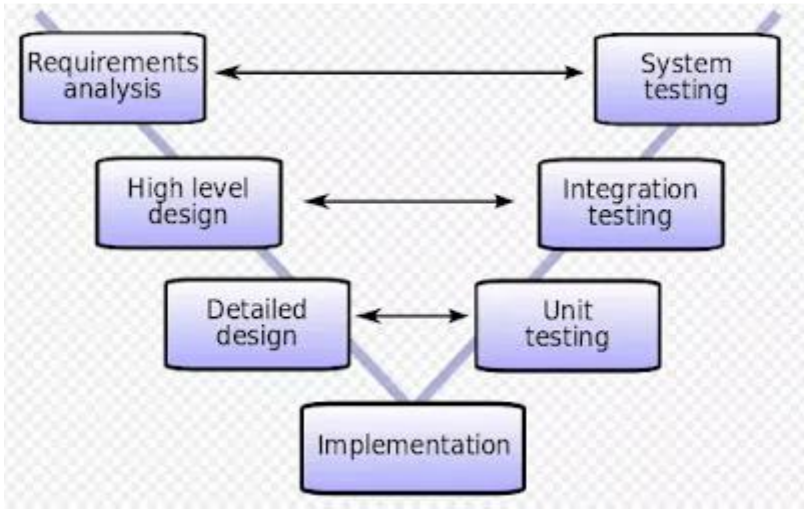
- Can be a costly model to use.

- Risk analysis needed highly particular expertise.
- Doesn't work well for smaller projects.

Please refer the below video for more detail on Spiral model:

V- Model

The V-model is a type of SDLC model where process executes in a sequential manner in V-shape. It is also known as Verification and Validation model. It is also known as Verification and Validation model. The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage.



V- Model

- The V-model is a type of SDLC model where process executes in a sequential manner in V-shape. It is also known as Verification and Validation model.
- The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage.
- This is a highly-disciplined model and the next phase starts only after completion of the previous phase.

When to use?

- Requirements are well defined, clearly documented and fixed.
- Product definition is stable.
- There are no ambiguous or undefined requirements.
- The project is short.

Advantages:

- Start testing from beginning of the project
- Testing include both validation and verification

- Defect found at early stages cost of fixing defect is less
- As testing is involve from beginning better understanding of requirement.

Disadvantages:

- High risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.
- Once an application is in the testing stage, it is difficult to go back and change a functionality.
- No working software is produced until late during the life cycle.

Please refer the below video for more detail on Verification and Validation:

Please refer the below video for more detail on V&V Model:

Agile Development Model

- Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process.
- All the team member should work together to deliver a quality product.
- Management is also involved continuously in this project.
- It is one of the most widely used process in software development world as it helps teams to deliver the work faster in small increments with less issues.

General terminologies used in Agile process

- Product Backlog – User stories(requirements are split into user stories) will be added here. For any sprint, user stories will be picked from product backlog and worked upon.
- Epic – It is the complete feature or requirement which needs to be implemented.
- User story – An Epic is split into user stories based on the complexity or work involved in order to develop it. User story is generally part of Epic. In case of small feature, it will be a single user story.
- Story points – It is a measurement to define the complexity of the user story from dev and QA perspective. This activity will be done in Sprint planning.
- Burn down chart – This is a pictorial graph representation of time and work. This is drawn with the help of the time spent on tasks along with the work. Team need to burn the hours on daily basis for the work accomplished against the tasks.

Agile Process:

- Customer satisfaction by rapid, continuous delivery of useful software Here in Agile process, we can monitor the project progress every day.
- Every release is implemented in iterations or sprints. Every iteration or sprint consists of 2 or 3 weeks in general. However, this sprint duration varies from project to project and also to sprint to sprint as well sometimes.
- Many meetings were involved in this Agile process to know the progress of the release in each sprint.

Agile Meetings:

- Sprint Planning meeting is conducted before the start of the sprint to discuss the user stories that can be covered in that particular sprint.
- Daily stand up/Scrum is conducted to know the work done by each in the team. Mainly the below things are discussed,
- What work is done yesterday
- What work is planned for today
- What work is planned for tomorrow
- Any impediments to complete the work

Agile Meetings:

- Sprint Review meeting is conducted at the end of the sprint to discuss the progress and where does the team stand for that sprint. However this is a optional meeting which is conducted based on the projects/companies as required.
- Sprint Retrospective meeting is conducted once a sprint is completed and next sprint got started. Following things will be discussed,
- What went well in the last sprint
- What didn't went well in the last sprint
- What can be done to improve or perform well in the ongoing sprint
- Defect Triage meeting is conducted to review and have a proper action on the defects. Dev team, testing team and sometimes the Product owner also will be involved in this meeting. Defects identified in the particular sprint will be triaged and assigned to respective team or deferred sometimes depending on the defects.
- Sprint Planning meeting is conducted before the start of the sprint where the user stories for that particular sprint and the timelines for the same would be planned.

Advantages:

- Customer satisfaction by rapid, continuous delivery of useful software.
- Working software is delivered frequently (weeks rather than months).
- Weekly release are expected if any large requirement are split so that development and testing is completed in the week scheduled.
- Customers, developers and testers constantly interact with each other.
- Daily cooperation between business people and developers.

Disadvantages:

- Sometimes in Agile methodology the requirement is not very clear hence it's difficult to predict the expected result.
- In few of the projects at the starting of the software development life cycle it's difficult to estimate the actual effort required.
- Because of the ever-evolving features, there is always a risk of the ever-lasting project.
- For complex projects, the resource requirement and effort are difficult to estimate.

Best Agile and Scrum Certifications:

- Certified Scrum Professional
- Certified Scrum Master
- Agile Certified Coach
- Agile Certified Practitioner
- SAFe 4 Agilest

Please refer the below video for more detail on Agile model:

Please refer the below video for more detail on SAFe Agile:

Please refer the below video for more detail on Shift Left Testing in Agile:

<u>Prototype Model</u>

- The prototyping model is a systems development method in which a prototype is built, tested and then reworked as necessary until an acceptable outcome is achieved.
- If customer is not having clear understanding of requirement a sample design is developed and present to the customer. based on feedback finalized the requirement.

When to use Prototype model:

- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
- Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model.

Advantages:

- Active involvement :Users are actively involved in development.
- Easy detection of missing functionality: errors can be detected in the initial stage of the software development process.
- Quick feedback: Quicker user feedback helps you to achieve better software development solutions.

- Flexibility : Encourages innovation and flexible designing.

Disadvantages:

- This model is costly and time consuming.
- It has poor documentation because of continuously changing customer requirements.
- There may be too much variation in requirements.

Please refer the below video for more detail on Prototype model:

Hybrid Model

The hybrid model is the combination of two or more primary (traditional) models and modifies them as per the business requirements.

Hybrid Model Examples:

- Spiral and prototype
- V & V and Prototype

Advantages:

- The hybrid model is highly flexible.
- In this model, the customer rejection is less because of the Prototype.
- It is easy to implement because it has the flexibility of synchronization.

Disadvantages:

- It does not follow the usual standards.
- Every hybrid model is different from each other.

Please refer the below video for more detail on Hybrid Model:

Derived/Customized Model

- Here we will take a basic model or single model and change it according to the customer business requirement or company standards is called as derived model / customized model.
- The most significant disadvantage of previous models (waterfall and spiral) is that there were lots of customer rejection that happens after the application was developed, and there was no involvement of the customers in between the project.
- Hence, they started the new approach, which is known as the prototype model

Example

- The spiral model is derived from the Water Fall Model.
- V - Model is also derived from Water Fall Model.
- Water Fall model is Known as Mother of all Models.

Difference between Verification and Validation

Verification Vs Validation:

Verification	**Validation**
Verification is the process of checking the documents (BRD, SRS), design, code and programs.	Testing and validation of actual product.
Are we building the product write? s/w meets the specification or not? Here we are checking whether we are developing right product or not.	Are we building the right product or not? s/w meets the requirement or not? We check whether the developed module is right or not.
Static Testing	Dynamic Testing
Methods – Inspection, Reviews and walk through.	Methods - Unit, integration, system, Acceptance, black box, functional and non-functional testing.
QA comes under verification.	QC comes under validation.
QA involved	Tester involved with the help of QA
It comes first	It comes later
It finds the bug early in the development life cycle.	It finds the bugs that verification cannot catch.

Quality Assurance VS Quality Control

Quality Assurance VS Quality Control

QA	**QC**
QA is the process of managing for quality.	QC is used to the verify the quality of the output.
QA aims to prevent the defect.	QC aims to identify and fix the defect.

It is method to manage the quality verification.	It is a method to verify the quality validation.
It is preventing technique.	It is a corrective technique.
It is the process to create the deliverables.	It is the process to verify the deliverables.
QA -> SDLC process.	QC-> STLC Process.
Involvement of whole team.	Involvement of Testing Team
Perform before QC.	After QA

Unit Testing and Its Techniques

What is Unit Or Component Testing :

- Unit testing is a software development process in which individual units of source code , sets of one or more computer program modules are tested to determine whether they are fit for use or not.
- Testing is conduct by developer or white box tester.
- To conduct this testing programming knowledge is required.
- This technique allows developers to analyze code, catch bugs earlier, and fix them faster.

Levels of Software Testing:

Advantages of Unit Testing

- Unit testing significantly improves code quality.
- Unit testing helps identify all kinds of issues with the software at a very early stage.

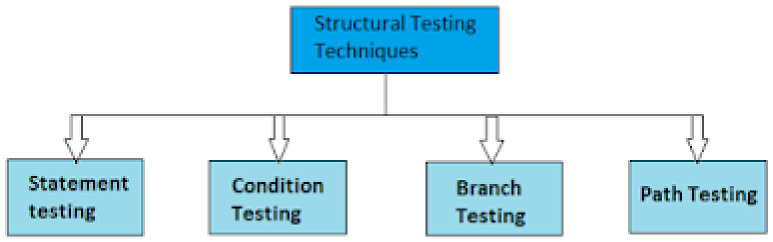
- If there are any problems, they are detected early on and making changes to the system thus becomes much easier.
- Reduce Costs

Unit Testing Techniques

- Structural Testing Techniques
- Functional Testing
- Error Based Testing Techniques

1. Structural Testing Techniques

Structure-based testing techniques use the internal structure of a software to derive test cases. It is a white box testing technique it required programming skills



Structure-based techniques can also be used at all levels of testing

- Statement Testing :- Testing each line in programming at least one time.
- Condition Testing:- Verify the conditional statement are working correct or not.
- Branch Testing:- Verify all the branch statement are working correct.
- Path Testing:- Testing the flow of the execution of the program.

2. Functional Testing

- FUNCTIONAL TESTING is a type of software testing whereby the system is tested against the functional requirements
- Functions (or features) are tested by feeding them input and examining the output.
- Functional testing ensures that the requirements are properly satisfied by the application.
- Functional testing is a quality assurance process and a type of black-box testing.

3. Error Guessing Testing Techniques

- Error Guessing is a Software Testing technique on guessing the error which can prevail in the code.
- It is a black box testing.
- Error guessing is a technique in which there is no specific method for identifying the error. It is based on the experience of the test analyst

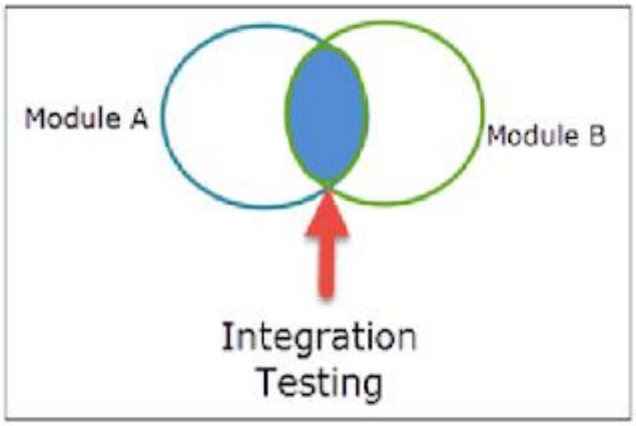
- Testing is conducted to analyses the impact of the defect on the application.

Integration Testing And Its Approaches

Integration Testing And Its Approaches

What is Integration Testing:

- Testing is conduct by integrating two or more component
- with in the same system or two different system are integrated.
- It is a phase in software testing in which individual software modules are combined and tested as a group.
- During the integration we verify the data communication between the component.
- We calculate the performance.(Time taken to complete the transaction)
- We verify the function of the application.



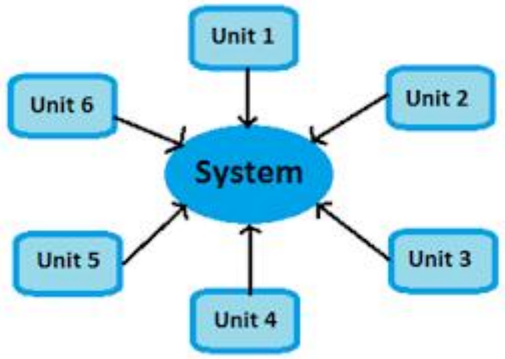
Integration Testing Approaches:

- Big - Bang Integration approach
- Incremental Approach

1. Top - Down Integration
2. Bottom – Up Integration
3. Mixed Or Sandwich integration

1. Big – Bang Integration Approach

- Testing is conducted on multiple component together.
- All the units are tested together as one single combined unit.
- This form of integration testing works well for smaller systems.



Limitations Faced in Big – Bang Integration Approach

- In this approach Detailed testing is not possible if any defect is found it will take time or effort to analysis the defect.
- It is also difficult to find the root cause of a defect in this method.
- Critical modules are not prioritized which could increase the overall risk.

2. Incremental Approach

- In Incremental integration testing, the developers integrate the modules one by one using stubs or drivers to uncover the defects. This approach is known as incremental integration testing.
- Testing, in which components or systems are integrated and tested as long as all components or systems are integrated and tested completely.

Types of incremental approach

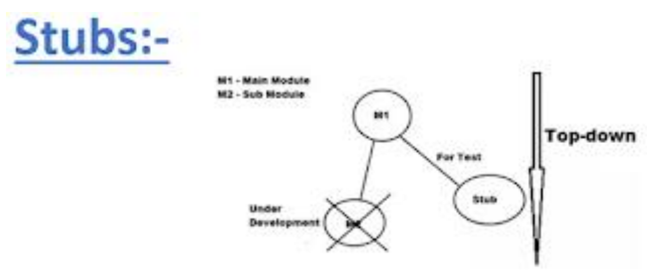
- Top - Down Integration:- Testing is conduct from main module to sub module. In this approach verify the data updated in main module is updated in sub module.
- Bottom – Up Integration:- Testing is conduct from sub module verify the data tested in sub module thus make sure that highest module is tested properly.
- Mixed Or Sandwich integration:- It is combination of top-down and bottom-up integration.

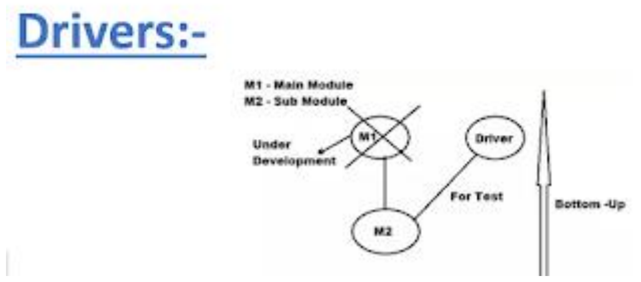
Stubs And Drivers

This are the temporary component which are used in place of sub module and main module

Stubs :- It is substitute for sub module in top - down integration approach.

Driver :- It is substitute for Main module in Bottom - up integration approach.





Advantages of Integration Testing

- It ensures that internal modules and components communicate properly.
- It performs regression testing on important connection points.
- It also helps to detect the issues related to the interface between modules.
- Integration testing helps to stimulate the interaction between various modules.
- It Covers multiple modules to provide broader test coverage.

Disadvantages of Integration Testing

- Difficult to perform – It is very difficult to perform as compared
- to system testing in which we can consider the application as a
- black box.
- Time-consuming – It is very time-consuming and
- resource-intensive to test all the interfacing between the different connected modules.
- Additional efforts – It requires the creation of stubs and drivers which if not created correctly can result in inadequate testing.

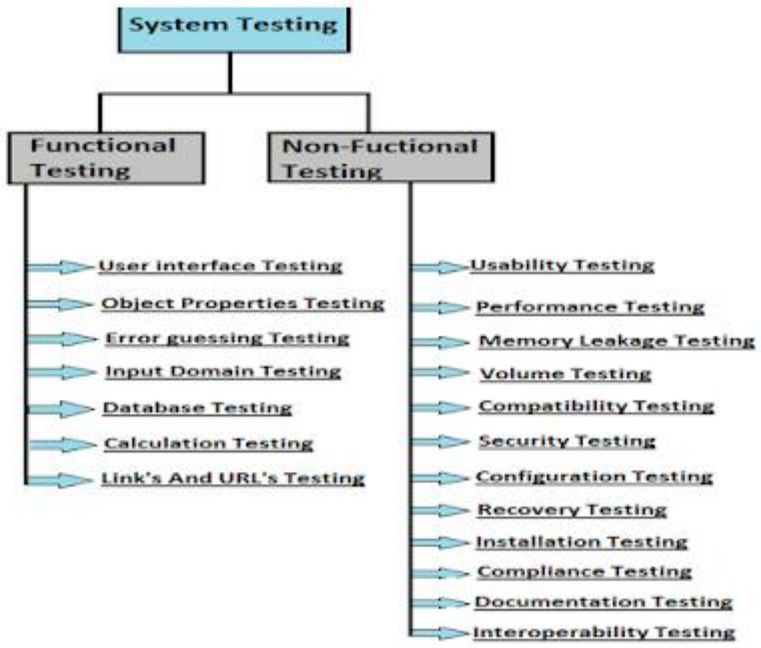
System Testing And Its Types

What is System Testing?

- System testing is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements.
- Testing is conducted by tester.
- It is a black box testing.
- This testing is conduct after the complete code is developed.
- During system testing we will conduct both functional and non-functional testing

Types of System Testing

- Functional Testing
- Non-Functional Testing



Advantages of System Testing

- It helps in getting maximum bugs before acceptance testing.
- It ensure the application is validated as per customer requirement. (SRS ,FRS)
- It is the first testing level in which the whole system is under test from end to end.
- This testing phase uses the test environment which is similar to the real business environment or production environment.

Functional Testing

These are black box testing techniques which test the functionality of the application.

- Functional testing validate the application's functionality is working as per requirement or not.
- It Is a quality assurance process.
- Functions are tested by feeding them input and examining the output.

Types of Functional Testing

- User Interface Testing
- Object Properties Testing

- Error Guessing Testing
- Input Domain Testing
- Database Testing
- Calculation Testing
- Link's and URL's Testing

Non-Functional Testing

- In this type of testing we check non-functional aspects (performance, usability, reliability, etc) of a software application.
- We test the characteristic and feature of an application.
- Non Functional testing has a goal to validate the performance of the software.

Types of Non-Functional Testing

- Usability Testing
- Performance Testing
- Memory leakage Testing
- Volume Testing
- Compatibility Testing
- Security Testing
- Configuration Testing
- Recovery Testing
- Installation Testing
- Compliance Testing

Acceptance Testing And Its Levels

What is Acceptance Testing:

- After testing is completed by tester before deploying the application into production business user will conduct testing to ensure the application is working as per there requirement or not called acceptance testing.
- This is a type of testing done by users, customers, or other authorized entities to determine application/software needs and business processes.
- Software testing where a system is tested for acceptability.

Advantages of Acceptance Testing

- Reduces the risk of defects being identified in production.
- It increases the satisfaction of clients as they test application itself.
- Increasing software robustness and usability.
- Increase end-user happiness

Levels of Acceptance Testing

- Alpha Test OR Pre UAT

- Beta Test OR Post UAT

Alpha Test OR Pre UAT Testing

- Alpha testing is conducted in the development company.
- Alpha tests are carried out early in the development process by internal staff ,project manager teams up with the developer to define specific goals for alpha testing.
- Testing is conduct in test environment.
- Testing is done by testers and quality analyst.
- Testing is conduct for application and product.

Advantages of Alpha Testing or Pre UAT

- It helps to uncover bugs that can pose a serious threat.
- It helps to deliver good quality software for beta testing
- During this testing, the real-time behavior of the users and the environment is imitated.

Beta Test OR Post UAT

- Testing is conducted at customer location.
- Beta testing is one of the final steps in your software development lifecycle (SDLC) before a product goes live.
- Beta Testing is performed by real users of the software application in real environment.
- Beta Testing allows a company to test post-launch infrastructure.

Advantages of Beta Testing or Post UAT

- Direct feedback from customers is a major advantage of Beta Testing.
- Reduces product failure risk via customer validation.
- Improves product quality via customer feedback.

Different type of software testing Method

1. White Box testing: (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing):
White box testing is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester.

Different Types and Techniques of white box testing
Unit Testing : The developer carries out unit testing in order to check if the particular module or unit of code is working fine. The Unit Testing comes at the very basic level as it is carried out as and when the unit of the code is developed or a particular functionality is built.
Static and dynamic Analysis: Static analysis involves going through the code in

order to find out any possible defect in the code. Dynamic analysis involves executing the code and analyzing the output.
Statement Coverage: In this type of testing the code is executed in such a manner that every statement of the application is executed at least once. It helps in assuring that all the statements execute without any side effect.
Branch Coverage: No software application can be written in a continuous mode of coding, at some point we need to branch out the code in order to perform a particular functionality. Branch coverage testing helps in validating of all the branches in the code and making sure that no branching leads to abnormal behavior of the application.
Security Testing: Security Testing is carried out in order to find out how well the system can protect itself from unauthorized access, hacking – cracking, any code damage etc. which deals with the code of application. This type of testing needs sophisticated testing techniques.
Mutation Testing: A kind of testing in which, the application is tested for the code that was modified after fixing a particular bug/defect. It also helps in finding out which code and which strategy of coding can help in developing the functionality effectively.

Advantages of White Box Testing:

- As the knowledge of internal coding structure is prerequisite, it becomes very easy to find out which type of input/data can help in testing the application effectively.
- Its helps in optimizing the code and in removing the extra lines of code, which can bring in hidden defects.

Disadvantages of White Box Testing:

- As knowledge of code and internal structure is a prerequisite, a skilled tester is needed to carry out this type of testing, which increases the cost.
- It is nearly impossible to look into every bit of code to find out hidden errors, which may create problems, resulting in failure of the application.

2. Black box testing: BLACK BOX TESTING, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

The purpose of black box testing
Black-box testing checks that the user interface and user inputs and outputs all work correctly. Part of this is that error handling must work correctly. It's used in functional and system testing..
Example : an operating system like Windows, a website like Google, a database like Oracle or even your own custom application. Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

Types of Black Box Testing:

There are several phases of which are segregated into different types:
Regression testing: Regression testing verifies that recent code changes haven't altered or destroyed the already existing functionality of a system. Regression testing examples include iteration regression and full regression, and both can be covered with manual and automated test cases.
Unit testing: UNIT TESTING is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.
Beta testing: Beta Testing is performed by real users of the software application in a real environment. Beta testing is one of the type of User Acceptance Testing. Beta version of the software, whose feedback is needed, is released to a limited number of end-users of the product to obtain feedback on the product quality.
Integration testing: INTEGRATION TESTING is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.
System testing: SYSTEM TESTING is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system.
Functional testing: FUNCTIONAL TESTING is a type of software testing whereby the system is tested against the functional requirements/specifications. Functions (or features) are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application.
Load testing: Load testing is a type of non-functional testing. A load test is type of software testing which is conducted to understand the behavior of the application under a specific expected load. Load testing is performed to determine a system's behavior under both normal and at peak conditions.

Advantages of Black Box Testing

- The test is unbiased because the designer and the tester are independent of each other.
- The tester does not need knowledge of any specific programming languages.
- The test is done from the point of view of the user, not the designer.
- Test cases can be designed as soon as the specifications are complete.

Disadvantages of Black Box Testing

- The test can be redundant if the software designer has already run a test case.
- The test cases are difficult to design.
- Testing every possible input stream is unrealistic because it would take a inordinate amount of time; therefore, many program paths will go untested.

3. Gray box testing: GRAY BOX TESTING is a technique to test the software product or application with partial knowledge of the internal workings of an application. The purpose of this testing is to search for defects due to improper code structure or improper functioning usage of an application
Example of Gray Box Testing would be when the codes for two units/modules are studied (White Box Testing method) for designing test cases and actual tests are conducted using the exposed interfaces (Black Box Testing method).

Gray-box testing Techniques:
Regression testing: Regression testing verifies that recent code changes haven't altered or destroyed the already existing functionality of a system. Regression testing examples include iteration regression and full regression, and both can be covered with manual and automated test cases.
Pattern Testing: A pattern tester is someone who tests a sewing pattern and provides feedback to the pattern designer before it is launched into the market . This process is done in order to check the sewing instructions, as well as the fit of the pattern on different body types, before the pattern is released to the public.
Orthogonal array testing: Orthogonal array testing. Orthogonal array testing is a black box testing technique that is a systematic, statistical way of software testing. It is used when the number of inputs to the system is relatively small, but too large to allow for exhaustive testing of every possible input to the systems.
Matrix testing: A matrix is a concise organizer of simple tests, especially useful for function tests and domain tests. It groups test cases that are essentially the same. For example, for most input fields, you'll do a series of the same tests, checking how the field handles boundaries, unexpected characters, function keys, etc

Advantages of Gray Box Testing:

- Gray-box testing provides combined benefits of both white-box and black-box testing
- It is based on functional specification, UML Diagrams, Database Diagrams or architectural view
- Gray-box tester handles can design complex test scenario more intelligently
- The added advantage of Gray-box testing is that it maintains the boundary between independent testers and developers.

Disadvantages of Gray Box Testing:

- In Gray-box testing, complete white box testing cannot be done due to inaccessible source code/binaries.
- It is difficult to associate defects when we perform Gray-box testing for a distributed system.

What is UI (user interface) testing?

What is UI (user interface) testing?

- UI Testing, also known as GUI (Graphical User Interface) Testing.
- Verify the page or window displayed or not.
- Verifying the objects displayed in the page.
- This usually means testing the visual elements on the application.

Why we do GUI testing?

- GUI testing is helps to deliver high-quality and user-friendly software. In the end, you achieve a higher level of user engagement and satisfaction.
- To ensure the images and other feature are user friendly or not.
- To Ensure that user will use the application further or not.

What we verify as part of GUI testing

- In GUI we mainly test the visible design element like button, icon, text box etc…
- We Check for the fonts are readable or not
- Check for the images. There clarity and pixel size and position.
- Check for the page if it is easy to understand as per customer point of view .
- Check for the buttons and boxes are Proper not overlapped.
- Check for grammar and spellings in page.
- Testing of the error messages that are getting displayed.
- Testing the different sections of the screen

Advantages of UI or GUI Testing

- Helps validate the compliance of various icons and elements with their design specifications.
- The sooner we can catch offending elements in our UI, the quicker they can be resolved.
- Offers developers and testers ease of use and learning.
- Efficiently reduces the number of risks towards the end of development life cycle.

What is Object Properties Testing?

- Verify the Properties of an object in the page like enable , disable , color , x-y coordinates etc .
- In this test, most likely, we need to check whether an object's property meets a specific condition or not.

Object Properties Testing example:

- In the yahoo or gmail inbox first page previous and 1st link should be disable , when go to next page previous and 1st link should get enable.

What is Error Guessing Testing Technique?

Error Guessing Testing

- Testing is conducted by performing invalid operations and validate the error massage is displaying or not .
- The Error massage should meaningful to understand.
- It is a type of testing method in which prior experience in testing is used to uncover the defects in software. It is an experience based test technique in which the tester uses his/her past experience or intuition to gauge the problematic areas of a software application.

Error Guessing Testing example:

- We need to test a program which read's a file. What happen if the program gets a file which is empty OR The file does not exist?
- Enter blank space into text fields
- Use max limits of files to be uploaded

Advantages of Error Guessing Testing

- It uncovers those defects which would otherwise be not possible to find out, through formal testing.
- To expand the test coverage.
- It does not follow any specific rules.

Disadvantages of Error Guessing Testing

- Only experienced testers can perform this testing. You can't get it done by freshers.
- It also cannot guarantee that the software has reached the expected quality benchmark.

Complete Article on Smoke and Sanity Testing

Smoke Testing:

- Smoke test is also known as Build Verification Testing OR Build Acceptance Testing.
- Smoke testing is the preliminary check of the software after a build and before a release.
- It is a type of acceptance testing, provide an initial check that a new software build and its critical functionality are stable or not.
- Smoke testing is a group of tests that are executed to verify if the critical functionalities of a particular build are working fine as expected or not.

How to do Smoke Testing:

- The development team deploys the build in QA. We will need to perform setup steps.

- Then the subsets of test cases are taken, and then testers run test cases on the build.
- The QA team test the application against the critical functionalities. These series of test cases are designed to expose errors that are in build.
- After the smoke test, we need to clean up. This might include stopping a server, deleting files, or emptying database tables. This could also be done before the initial setup step to ensure that the environment is clean before any tests are started.

Why We Perform Smoke Testing:

- Smoke Testing is done whenever the new functionalities of the software are developed and integrated with the existing application.
- It ensures that all critical functionalities are working correctly.
- It reveal simple failures but severe enough to , reject a prospective software release.
- It helps to find issues in the early phase of testing.

Smoke Testing Example

A web page for insurance add a claim status page. Tester would apply smoke test to verify that the existing build works on fundamental level , such as user can login in or not and navigate to claim status page or not , and retrieve the status report of specific claim without the app crashing.

Sanity Testing:

- Sanity Testing is a subset of regression testing.
- After receiving the software build, sanity testing is the 1st test to ensure that the code changes introduced are working as expected.
- If any defect found in test we can reject the build and stop the testing.
- After Sanity test complete the status either Pass Or Fail has to be reported to the developers.
- The test is conducted for very short period of time (15-30 min Max)

How to do Sanity Testing

- Unlike other types of testing, sanity does not need a handful of techniques, You do not need to write test cases for sanity testing because you want to perform quick and speedy testing.
- The first thing in Sanity Test is to identify the new functionalities, changes or any bug fixes.
- Then we will check the newly implemented changes if they are working fine or not .
- At Last we will randomly check a few related functionalities to see if they are also working as expected.

- If it all goes good, then the release can be passed for further testing or if any one functionality is not working as per requirement or got failed we can reject the build and stop the testing.

Why We Perform Sanity Testing

- It is also known as Surface Level Testing, this must be done to quickly evaluate the quality of regressions made to the software.
- Sanity is done to determine if new module additions to an existing software build are stable enough to pass to the next level of testing or not.
- If sanity test is pass we can continue further testing, if its fail we can reject the build and stop the testing so developers and fix the changes ASAP.

Sanity Testing Example

A web page for loan provider returns a 404 error for personal loan page. The developers fix the issue and submit the build for testing. The QA professional perform a sanity test to determine whether the basic functionality and navigation for that specific page is working as intended or not.

Smoke Vs Sanity

Smoke	Sanity
Smoke Testing is performed to ascertain that the critical functionalities of the program are working fine.	Sanity testing is done at random to verify that each functionality is working as expected.
Smoke testing is usually documented and scripted.	Sanity testing is not documented and is unscripted.
Smoke Testing has a goal to verify “stability”.	Sanity Testing has a goal to verify “rationality”.
Testing is done by both developers or testers.	Sanity Testing is done by only testers.
Smoke testing is a subset of acceptance testing.	Sanity testing is a subset of Regression Testing.
This is a wide and High Level testing.	This is a wide and shallow testing.
It is a well elaborate and planned testing.	This is not a planned test and is done only when there is a shortage of time.
Smoke testing exercises the entire system from end to end.	Sanity testing exercises only the particular component of the entire system.

What is Positive and Negative Testing?

Positive Testing

- Positive Testing is a type of testing which is performed on a software application by providing the valid data as an input.
- Positive testing is a type of software testing that is performed by assuming everything will be as expected.
- It is performed with the assumption that only valid and relevant things will occur.
- In this type, testing performed within the boundaries and this testing checks that the product /application is behaving as per the specification document with a valid set of test data.

Negative Testing

- Negative testing is a method of testing an application or system that ensures that the plot of the application is according to the requirements and can handle the unwanted input and user behavior.
- It is also known as error path testing or failure. And it helps us to identify more bugs and enhance the quality of the software application under test .
- Negative testing uses invalid input data, or undesired user behaviors, to check for unexpected system errors.
- We can say that the negative testing is executing by keeping the negative point of view in simple terms.

Difference between positive testing and negative testing:

Positive Testing	**Negative Testing**
Positive Testing means testing the application or system with valid data.	Negative Testing means testing the application or system with invalid data.
It is always done to verify the known set of test conditions.	It is always done to break the project or product with unknown set of test conditions.
It ensures software is normal.	It ensures 100% defect free software.
It doesn’t cover all possible cases.	It covers all possible cases.
It can be performed by people having less knowledge.	It can be performed by professionals.
Positive testing is implemented only for the expected conditions.	Negative testing is implemented only for unexpected conditions.
It is less important as compare to negative testing.	It is more important as compare to positive testing.
Positive testing can be implemented on every application.	Negative testing can be implemented when the possibilities of unpredicted conditions.

What is Exploratory Testing?

Exploratory Testing?

- Exploratory testing is an approach to software testing that is concisely described as simultaneous learning, test design and test execution.
- Exploratory testing allows you to think outside the box and come up with scenarios that might not be covered in a test case.
- In simple word it is a type of software testing where Test cases are not created in advance but testers check system on the fly.
- It focuses on discovery and relies on the guidance of the individual tester to uncover defects that are not easily covered in the scope of other tests.
- It doesn't restrict the tester to a predefined set of instructions.
- This is a test approach that can be applied to any test technique, at any stage in the development process.

When should we do Exploratory Testing?

- In many software cycles, an early iteration is required when teams don’t have much time to structure the tests. Exploratory testing is quite helpful in this scenario.
- When testing mission-critical applications, exploratory testing ensures you don’t miss edge cases that lead to critical quality failures.
- It is especially useful to find new test scenarios to enhance the test coverage.
- It helps review the quality of a product from a user perspective.

Advantages of Exploratory testing

- It requires limited preparation, which allows you to save time and quickly jump to execution.
- In Exploratory Testing, you can generate your own test scenarios on the fly, which will allow you to dive deeper into the functional aspects of the product.
- This test is much less time-consuming.
- It allows you to think outside the box and come up with scenarios that might not be covered in a test case.
- This allows the tester to find defects that are beyond the scope of the listed scenarios.
- This is an approach that is most useful when there are no or poor specifications and when time is severely limited.

Disadvantages of Exploratory testing

- In exploratory testing, once testing is performed it is not reviewed.
- Difficult to document each procedure.
- It is difficult to reproduce the failure.
- It is not easy to say which tests were already performed.
- Limited by testers’ skills set.

- Reporting is difficult without planned scripts.

Retesting and Regression Testing

Re-testing

- Testing the same functionality with different combination of input data is called retesting.
- Retesting essentially means to test something again.
- In simple words, Retesting is testing a specific bug after it was fixed.
- Retesting ensures that the issue has been fixed and is working as expected.
- It is a planned testing with proper steps of verification
- In some cases the entire module is required to be re-tested to ensure the quality of the module.

Why and when we perform Retesting

- Retesting is used when there is any specific error or bug which needs to be verified.
- It is used when the bug is rejected by the developers then the testing department tests whether that bug is actual or not.
- It is also used to check the whole system to verify the final functionality.
- It is used to test even the entire module or component in order to confirm the expected functionality.

Advantages

- Retesting ensures that the issue has been fixed and is working as expected.
- It improves the quality of the application or product
- It requires less time for verification because it’s limited to the specific issue or any particular feature.
- If the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively.

Disadvantages

- It requires new build for verification of the defect.
- Once the testing is started then only the test cases of retesting can be obtained and not before that.
- The test cases for re-testing cannot be automated.

Regression Testing

- If any changes are done in the existing build this test is conduct on the modified build to verify the changes are working correctly or not and because of this changes there are no side effect.
- In Regression Test the change functionality + dependent functionality are tested .

- The purpose of the regression testing is to find the bugs which may get introduced accidentally because of the new changes or modification.
- This also ensures that the bugs found earlier are NOT creatable.
- This helps in maintaining the quality of the product along with the new changes in the application.

When To use Regression Testing

- Any new feature is added to an existing feature.
- The codebase is fixed to solve defects.
- Any bug is fixed
- Changes in configuration.

Advantages

- It helps the team to identify the defects and eliminate them earlier in the software development life cycle.
- It ensures continuity of business functions with any rapid change in the software.
- Regression testing can be done by using the automation tools.
- It helps us to make sure that any changes like bug fixes or any enhancements to the module or application have not impacted the existing tested code.

Disadvantages

- If regression testing is done without using automated tools then it can be very tedious and time consuming because here we execute the same set of test cases again and again.
- Regression testing has to be performed for every small change in the code as even a small portion of code can create issues in the software.
- It takes time to complete the tests and this slows down the agile velocity.
- It's expensive and the cost is hard to justify.

Re-testing vs Regression Testing:

Re-Testing	**Regression Testing**
Retesting is about fixing specific defects that you've already found.	Regression testing is about searching for defects.
Retesting is done only for failed test cases.	Regression testing is performed for passed test cases.
Retesting is used to ensure the test cases which failed in last execution are fixed.	Regression testing is to ensure that changes have not affected the unchanged part of product.
Verification of bugs are included in the retesting.	Verification of bugs are not included in the regression testing.
Retesting is of high priority so it’s done before the regression testing.	Regression testing can be done in parallel with retesting.

Retesting the test cases cannot be automated.	Regression testing test cases can be automated.
In case of retesting the testing is done in a planned way.	In case of regression testing the testing style is generic.
Test cases of retesting can be obtained only when the testing starts.	Test cases of regression testing can be obtained from the specification documents and bug reports

What is Adhoc Testing? Difference between Adhoc testing and Exploratory Testing

Adhoc Testing

- When a software testing performed without proper planning and documentation, it is said to be Adhoc Testing.
- Ad hoc Testing is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects.
- Ad hoc testing is performed without a plan of action and any actions taken are not typically documented.
- Ad Hoc Testing implies learning of the software before its testing.
- Ad hoc Testing is done by executing the random scenarios and it is a form of negative testing which ensures the perfection of the testing.

Why do we do adhoc testing?

- The main aim of ad hoc testing is to find any defects through random checking.
- This can uncover very specific and interesting defects, which are easily missed when using other methods.
- Ad hoc testing can be performed when there is limited time to do exhaustive testing and usually performed after the formal test execution.
- Ad hoc testing will be effective only if the tester has in-depth understanding about the System Under Test.

Advantages:

- This method is very simple and can be performed without any training.
- It can be used when time period is limited.
- This can uncover very specific and interesting defects, which are easily missed when using other methods.
- This testing can be performed any time during Software Development Life Cycle Process (SDLC)

Disadvantages:

- This method is not recommended, when more scientific methods are available.

- The actual testing process is not documented since it does not follow a particular test case.
- It is difficult for the tests to regenerate an error in ad hoc testing.

Adhoc testing Vs Exploratory Testing

Adhoc Testing	**Exploratory Testing**
Ad Hoc Testing implies learning of the software before its testing.	Exploratory Testing, you learn and test the software simultaneously.
Documentation is not a basic need of this type of testing. The QA team always attends the testing without specific documentation.	Documentation is mandatory in Exploratory Testing. To assure the quality it's necessary to documents the detail of the testing.
Ad hoc is about the perfection of the testing.	Exploratory Testing is more about the learning of the application.
Ad hoc Testing helps to find innovative ideas from the research.	It helps to develop the application.
Ad hoc is a technique of testing an application; this provides a significant role in the software Production.	This is an approach of testing that combines the learning test results and creates a new solution.
It mostly works on the business concerns and increases the knowledge about the application.	It categorizes the problems and compare them from the problems found in past. This helps to reduce the time consumption.
Adhoc testing is not important to execute by an expert software testing engineer.	This always needed to be done by expert.
It works on negative testing mostly.	This testing works on positive testing niche.

Monkey and Gorilla Testing

Monkey Testing

- Monkey testing is a type of software testing in which a software or application is tested using random inputs with the sole purpose of trying and breaking the system.
- Monkey testing tests the whole system, There are no rules in this type of testing.
- Monkey testing is usually implemented as random, automated unit tests.
- Monkey testing is also known as Random testing, Fuzz Testing or Stochastic Testing.

- Monkey testing is a crucial testing method that is done to authenticate the functionality of the product application.
- Monkey testing is a kind of black box testing.

Gorilla Testing

- Gorilla testing is a software testing technique that repeatedly applies inputs on a module to ensure it is functioning correctly and that there are no bugs.
- Gorilla testing is a manual testing procedure and is performed on selected modules of the software system with selected test cases.
- Gorilla testing is not random and is concerned with finding faults in a module of the system.
- Gorilla Testing is also known as Torture Testing, Fault Tolerance Testing or Frustrating Testing.
- Gorilla Testing is a manual testing and it is repetitively performed.
- Gorilla testing is a kind of white box testing.

Difference between monkey and gorilla testing:

Monkey	**Gorilla**
Monkey testing is a type of software testing which is performed based on some random inputs without any test cases and checks the behaviour of the system and confirms whether it crashes or not.	Gorilla Testing is a type of software testing which is performed on a module based on some random inputs repeatedly and checks the module's functionalities and confirms no bugs in that module.
In Monkey Testing, no test case is used to test the application as it is a part of random testing.	It is performed repeatedly as it is a part of manual testing.
The Monkey Testing approach is primarily used in System Testing.	The Gorilla Testing approach is mainly used in Unit Testing.
Monkey testing is implemented on a whole system.	Gorilla testing is implemented on few selective components of the system.
No software knowledge is required in order to execute the monkey testing.	It requires minimum software knowledge in order to execute the gorilla testing.
The main objective of Monkey Testing is to check whether system crashes or not.	The main objective of Gorilla testing is to check whether the module is working properly or not.
Monkey testing is also known as Random testing, Fuzz Testing or Stochastic Testing.	Gorilla Testing is also known as Torture Testing, Fault Tolerance Testing or Frustrating Testing.
There are three types of Monkey Testing i.e. Dumb Monkey Testing, Smart Monkey Testing and Brilliant Monkey Testing.	While there is no such different types of Gorilla Testing available.

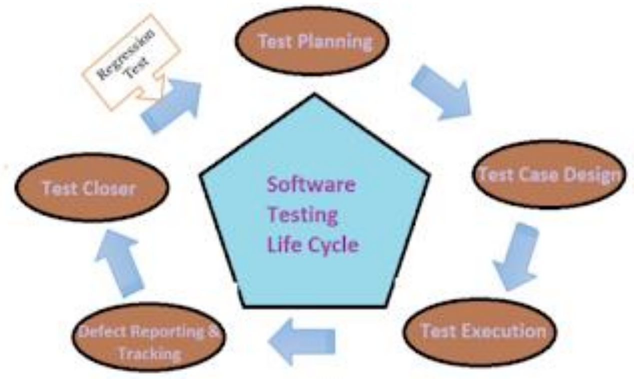
The implementation of Monkey testing does not require any planning or preparation.	The Gorilla testing cannot implement without any preparation or planning.

What is Software Testing Life Cycle?

The Software Testing Life Cycle (STLC) is a term that refers to the process of testing software. The Software Testing Life Cycle (STLC) is a set of actions that are carried out during the testing process to guarantee that software quality objectives are satisfied.

Any Project has to follow the below phases from beginning of a project to end of the project in Testing :

- Requirement Analysis
- Test Planning
- Test Case Design
- Test Execution
- Defect Reporting And Tracking
- Test Closer



1. Requirement Analysis :- It is tester's responsibility

- Study the requirements
- Identify types of tests to be performed
- Prepare RTM
- Automation feasibility analysis

2. Test Planning :- It is test lead's responsibility

- Understand the project
- Scop of testing
- Identify the resource
- Schedules
- Deliverables

- Approach
- Effort Estimations

3. Test Case Design :- It is Tester's Responsibility

- Design/Script the Test cases
- Review testcases
- Update test cases based on review and baseline test cases
- Create test data

4. Test Execution :- It is tester's responsibility

- Setup test environment
- Execute tests as per plan
- Document test results

5. Defect Reporting and Tracking:- It is Tester's Responsibility

- Report defect to developers
- Map defects to test cases in RTM
- Re-test the defect fixes
- Track the status of defect to closure

6. Test Closer :- It is tester and Test Lead's responsibility

Stop testing based on :-

- All functionalities are tested
- No new defect are found
- Schedule are conducted
- No risk in the project
- Test Closure Report

What is Test Planning and How to Create Test Plan?

Test Planning:

- Initially when we get the project it is responsibility for a test lead or senior tester to prepare the test plan for the project.
- A test plan is the foundation of every testing effort.
- A Test Plan refers to a detailed document that catalogs the test strategy, objectives, schedule, estimations, deadlines, and the resources required for completing that particular project.
- A good test plan clearly defines the testing scope and its boundaries. You can use requirements specifications document to identify what is included in the scope and what is excluded.

- By creating a clear test plan all team members can follow, everyone can work together effectively.

The steps to prepare the test plan :-

- Criteria
- Prepare test plan
- Get approval

1. Criteria:-

- To prepare test plan the test lead /senior tester should have the below details:
- Development schedule
- Project release date
- Services to be provided to customer
- Understand the project requirement
- Understanding of the complexity of the project
- Scop of testing
- No. of resources required
- Type of testing to be conducted

2. Prepare Test Plan :

- A Test Plan is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product.
- The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.

Test Plan document has the below details :

- Introduction
- Features to be tested (In scope)
- Features not to be tested (Out of scope)
- Pass/ Fail criteria
- Entry and Exit criteria
- Approach
- Schedule
- Test factor
- Recourses
- Training plans
- Configuration management
- Test Deliverable
- Test Environment
- Risk and Mitigation Plan
- Approval

3. Prepare Test Plan : Once test plan is ready it should be approved from test manager

Lets talk about different sections on a test plan in detail:

Introduction :- Specify the purpose / Requirement of the project.

Feature to be tested :- Specify the requirement which is need to be tested.

Feature not to be Tested :- Specify the requirement which is out of the scope of testing. We dose not need to test.

PASS / FAIL Criteria (Suspension Criteria):- If the below criteria is satisfied we can suspend the build from testing:

- The build is not deployed
- The sanity test is fail
- The build is unstable
- Wrong build is given

(Note: If the build is suspended the developer have to immediately resolve the issue and give the modified build)

Entry and Exit Criteria:- Specify the condition to start and stop the testing . for every level of testing there is entry and exit criteria

For example :-

System testing Entry Criteria :-

- Integration testing should be completed
- All the major defect found in the integration test should be fixed
- The test environment should be ready , Required access should be provided
- The Development should be completed (All Coding part)

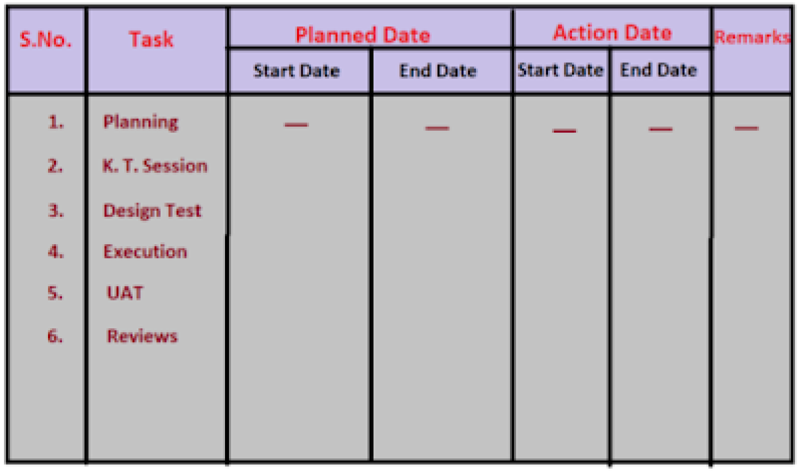
System testing Exit Criteria :-

- All the functionalities are tested
- No new defect are found
- No risk in the project
- All the defects are fixed
- The testing scheduled are completed

Approach :- Specify the process followed in the project like

Planning ◊ Design ◊ Execution ◊ Reporting

Schedules:- Specify the testing . The testing scheduled are prepared based on development schedules.



S.No.	Task	Planned Date		Action Date		Remarks
		Start Date	End Date	Start Date	End Date	
1.	Planning	—	—	—	—	—
2.	K. T. Session					
3.	Design Test					
4.	Execution					
5.	UAT					
6.	Reviews					

Resources :- Specify the list of team, resources and there responsibilities like testing team, development team, project managers etc.

Training plans :- Specify any training plan is required for the resources.

Configuration Management :- Specify hoe to manage all the project related documents , codes, testing documents etc.so these documents are accessible to all the team members.

There are tools are available like :

VSS◊ Visual source safe

CVS ◊ Concurrent version system for configuration management.

The tool provide security, version control (Each time the documents are updated a new version is created)

Test Deliverables :- Specify the list of documents to be submitted to the customer during testing or after testing Documents are like :

- Test Plan
- Test Case documents
- Test execution result
- Defect report
- Review report
- Test summary report etc.

Test Environment :- Specify the Hard ware configuration details, the additional software required in the test environment to access the application and conduct the testing.

Risk And Mitigation Plan:- Specify the challenges in the project and the solution to resolve the challenges.

The different risks are :

- Required documents are not available
- Estimations are not correct
- Delay in project deliverables
- Lack of skilled resources
- Lack of team coordination's ETC.

Approval :- After preparation of the plan the project manager and customer have to review and approve the plan.

Note :- During the planning phase it is the responsibility of test lead to recruit the tester based on the skills required for the project .

Test Strategy Vs Test Plan

Test Strategy:

- A test strategy is an outline that describes the testing approach of the software development cycle.
- Test strategy is a set of guidelines that explains test design and determines how testing needs to be done.
- The creation and documentation of a test strategy should be done in a systematic way to ensure that all objectives are fully covered and understood by all.
- It should also frequently be reviewed, challenged and updated as the organization and the product evolve over time.

Test Strategy Vs Test Plan:

Test Plan	**Test Strategy**
A test plan for software project can be defined as a document that defines the scope, objective, approach and emphasis on a software testing effort.	Test strategy is a set of guidelines that explains test design and determines how testing needs to be done.
Test plan include- Introduction, features to be tested, test techniques, testing tasks, features pass or fail	Test strategy includes- objectives and scope, documentation formats, test processes, team reporting structure, client communication strategy, etc.

criteria, test deliverables, responsibilities, and schedule, etc.	
Test plan is carried out by a testing manager or lead that describes how to test, when to test, who will test and what to test	A test strategy is carried out by the project manager. It says what type of technique to follow and which module to test
Test plan narrates about the specification.	Test strategy narrates about the general approaches.
It is done to identify possible inconsistencies in the final product and mitigate them through the testing process.	It is a plan of action of the testing process on a long-term basis.
It is used on a project level only.	It is used on an organizational level.
Test plan can change.	It is used by multiple projects and can be repeated a lot of times.
It is used by one project only and is very rarely repeated.	Test strategy cannot be changed.
Test planning is done to determine possible issues and dependencies in order to identify the risks.	It is a long-term plan of action. You can abstract information that is not project specific and put it into test approach

All you need to know about test cases

Test Case Design

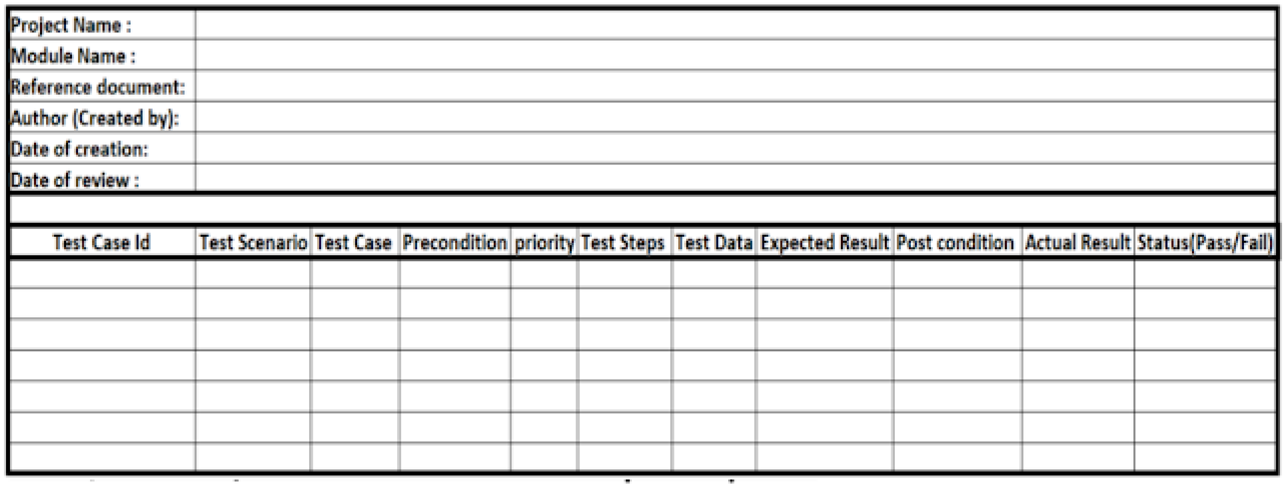
- Test case design refers to how you set-up your test cases. It is important that your tests are designed well, or you could fail to identify bugs and defects in your software during testing.
- It is responsibility of a tester to design the test cases.
- To design the test cases tester should :
- Understand the requirement
- Identify the scenario
- Design the test case
- Review the test case
- Update the test case based on review

How to create good test cases

- Test Cases need to be simple steps, transparent and easy to understand.
- Each and every test case should be traceable and it should be linked to "Requirement ID".
- Test cases should have only necessary and valid steps(Brief And Short) .
- Implement Testing Techniques – Positive And Negative Test Cases
- Test cases should be written in such a way that it should be easy to maintain.

- Test cases should cover the usability aspects in terms of ease of use.
- Different set of test cases should be prepared for the basic performance testing like multi-user operations and capacity testing.
- Test cases for Security aspects should be cover for example user permission, session management, logging methods.

Test cases Design Format:



Project Name :	
Module Name :	
Reference document:	
Author (Created by):	
Date of creation:	
Date of review :	

Test Case Id	Test Scenario	Test Case	Precondition	priority	Test Steps	Test Data	Expected Result	Post condition	Actual Result	Status(Pass/Fail)

Follow below steps to create Test cases

Test Case Id : It is a unique number to represent each test cases. It's good practice to follow some naming convention for better understanding and discrimination purposes. Like Tc_proj_module_number(001)

For example: Tc_Yahoo_Inbox_001

Test Scenario : A Test Scenario is defined as any functionality that can be tested. It is a collective set of test cases which helps the testing team to determine the positive and negative characteristics of the project. Test Scenarios are derived from test documents such as BRS and SRS.

Test Cases : A Test Case is a set of actions executed to verify a particular feature or functionality of your application. Test Cases are focused on what to test and how to test. Test Case is mostly derived from test scenarios.

Precondition : The condition to be satisfied to test the requirement. Conditions that need to meet before executing the test case.

Priority : The importance of the test case is called priority. It is a parameter to decide the order in which defects should be fixed. Priority means how fast defect has to be fixed.

Test Steps : Test Steps describe the execution steps. To execute test cases, you need to perform some actions. Each step is marked pass or fail based on the comparison result between the expected and actual outcome.

Test Data: It is data created or selected to satisfy the execution preconditions and inputs to execute one or more test cases. We need proper test data to execute the test steps.

Expected result : The output result as per customer requirement (As per SRS or FRS). The result which we expect once the test cases were executed.

Post Condition : Conditions that need to achieve when the test case was successfully executed.

Actual Result : The output result in the application once the test case was executed. Capture the result after the execution. Based on this result and the expected result, we set the status of the test case.

Status : The status as Pass or Fail based on the expected result against the actual result. If the actual and expected results are the same, mention it as Passed. Else make it as Failed. If a test fails, it has to go through the bug life cycle to be fixed.

Other important fields of a test case template:

Project Name: Name of the project the test cases belong to.

Module Name: Name of the module the test cases belong to.

Reference Document: Mention the path of the reference documents (if any such as Requirement Document, Test Plan , Test Scenarios, etc.,)

Author (Created By) : Name of the Tester who created the test cases.

Date of Creation: When the test cases were created.

Date of Review: When the test cases were reviewed.

Reviewed By: Name of the Tester who created the test cases.

Executed By: Name of the Tester who executed the test case.

Date of Execution: When the test case was executed.

Comments: Include value information which helps the team.

Test Design Testing Techniques

Test Design Testing Techniques :

- BAV (Boundary Value Analysis)
- ECP (Equivalence Class Partition)
- Decision Table Based Testing
- State Transition
- Error Guessing Testing

1. BAV (Boundary Value Analysis):

- The technique specify to validate the boundary validation.
- It is based on testing at the boundaries.
- We will be testing both the valid and invalid input values in the BVA.

Conditions are like :

Min	Max
Min-1	Max-1
Min+1	Max+1

The above 6 conditions are enough.

BAV (Boundary Value Analysis) Example:

Password field accept 6-32 character then we only test for :-

Min : 6	Max : 32
Min-1 : 5	Max-1 : 31
Min+1 : 7	Max+1 : 33

These 6 conditions are enough for password field testing.

2. ECP (Equivalence Class Partition):

- The technique specify to test for valid and invalid combination.
- The idea behind this technique is to divide a set of test conditions into groups or sets that can be considered the same.
- It is a black-box testing technique or specification-based testing technique in which we group the input data into logical partitions called equivalence classes.

ECP Example:

Any mail id field accepts alpha and numeric data :

Valid Data	Invalid Data
A – Z	All special character
a – z	
0 – 9	

If a Text field accept 1000 -1500 the partition should be:

Valid Data	Invalid Data
1000 – 1500	a – z
	A – Z
	Numbers < 1000
	Numbers > 1500
	All special character

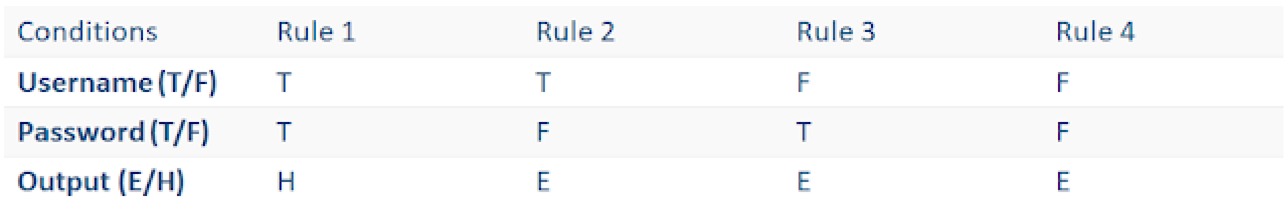
3. Decision Table Based Testing:

- Decision table testing is a software testing technique used to test system behavior for different input combinations.
- In this methodology the various input combinations and the accompanying system behavior (Output) are tabulated.
- This is a systematic approach.
- Decision Table Testing is a Black Box test design technique (behavioral or behavior-based technique).
- When a system has complex business rules, then the decision table testing technique helps in identifying the correct test cases.

Example: How to Create a Login Screen Decision Base Table

Let's make a login screen with a decision table. A login screen with User id and Password Input boxes and submit button.

The condition is simple if the user provides correct username and password the user will be redirected to the homepage. If any of the input is wrong, an error message will be displayed.



Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Username (T/F)	T	T	F	F
Password (T/F)	T	F	T	F
Output (E/H)	H	E	E	E

T – Make sure to use Correct username/password

F – Incorrect username/password

E – An error message is displayed

H – The home screen is displayed

Interpretation:

Case 1 – Username and password both were correct, and the user navigated to homepage

Case 2 – Username was correct, but the password was wrong. The user is shown an error message.

Case 3 – Username was wrong, but the password was correct. The user is shown an error message.

Case 4 – Username and password both were wrong. The user is shown an error message.

While converting this to test case, we can create 2 scenarios :

First one :

Enter correct username and correct password and click on Submit button, and the expected result will be the user

Should be navigated to homepage.

Second one from the below scenario:

- Enter Correct username and incorrect password and click on Submit button, and the expected result will be the user should get an error message.
- Enter incorrect username and incorrect password and click on Submit button, and the expected result will be the user should get an error message.
- Enter incorrect username and correct password and click on Submit button, and the expected result will be the user should get an error message.

As they essentially test the same rule.

4. State Transition:

- State transition testing helps to analyze behavior of an application for different input conditions.
- In this test outputs are triggered by changes to the input conditions or changes to 'state' of the system.
- In other words, tests are designed to execute valid and invalid state transitions.
- Testers can provide positive and negative input test values and record the system behavior. It is the model on which the system and the tests are based.
- It is a black box testing which is used for real time systems with various states and transitions involved.

When to Use State Transition?

- This can be used when a tester is testing the application for a finite set of input values.
- When we have sequence of events that occur and associated conditions that apply to those events
- This will allow the tester to test the application behavior for a sequence of input values.

When to Not Rely On State Transition?

- When the testing is not done for sequential input combinations.
- If the testing is to be done for different functionalities like exploratory testing.

Example:

Let’s consider an Login page function where if the user enters the invalid password three times the account will be locked.

In this system, if the user enters a valid password in any of the first three attempts the user will be logged in successfully. If the user enters the invalid password in the first or second try, the user will be asked to re-enter the password. And finally, if the user enters incorrect password 3rd time, the account will be blocked.

	Correct Password	Incorrect Password
S1) Start	S5	S2
S2) 1st attempt	S5	S3
S3) 2nd attempt	S5	S4
S4) 3rd attempt	S5	S6
S5) Access Granted	–	–
S6) Account blocked	–	–

In the table when the user enters the correct PIN, state is transitioned to S5 which is Access granted. And if the user enters a wrong password he is moved to next state. If he does the same 3rd time, he will reach the account blocked state.

5. Error Guessing Testing:

- Testing is conducted by performing invalid operations and validate the error massage is displaying or not .
- The Error massage should meaningful to understand.
- It is a type of testing method in which prior experience in testing is used to uncover the defects in software. It is an experience based test technique in which the tester uses his/her past experience or intuition to gauge the problematic areas of a software application.

Example:

- We need to test a program which read's a file. What happen if the program gets a file which is empty OR The file does not exist???
- Enter blank space into text fields
- Use max limits of files to be uploaded

Test Case Review Process and Guidelines

Test Case Review :

- Test case review process is an important process to follow in software testing. Test case should be effective and also follow the standards to write test case.
- After preparation of the test cases review are conduct to ensure the completeness and correctness of the test case, proper flow, and maximum test coverage.
- The test lead or project manager have to approve the test cases.
- During this review process, if the reviewer found any mistake, he/she writes it in a separate document, which is known as Review document and sends it back to the author.

Test Case Review Techniques :

- Self-review: This is carried out by the tester who wrote the test cases. By looking at SRS/FRD, he/she can see if all of the requirements are met or not.
- Peer review: Before test lead the test cases reviewed by another team member who is not familiar with the system is called peer review. Maker and Checker are two other names for the same thing.
- Supervisory review: This is done by a team lead or project manager who is higher in rank than the tester who wrote the test cases and has an extensive understanding of the requirements and system.

Common mistakes which are check during Test Case Review :

- Incomplete Test Cases. Missing negative test cases. No Test Data. Inappropriate/Incorrect Test data.
- Spelling errors can sometimes cause a lot of confusion or make a statement difficult to grasp.
- While review process, it is very important to check whether all the standards and guideline are properly followed.
- If proper template is followed then it becomes easy to add/modify test cases in future and test case plan looks organized.
- If the grammar is incorrect, the test case can be misinterpreted, leading to incorrect findings.
- Test cases should have a very simple language which is easy to understand.
- Replication: It refers to the duplicate test cases removal. It is possible that two or more test cases test the same thing and can be merged into one, this would save time and space.
- Test case which are uselessness due to change in requirements or some modifications. Such test cases must be removed.

Test Case Repository :

- A test case repository is a centralized location where all the baseline test cases (written, review, and approved) are stored.
- Test repository comprises test cases that encompass all key possibilities of permutations and combinations in workflow execution and transaction, ensuring that all variations in system and user interactions are covered.
- A test case repository is used to store the approved test cases only.
- Any test engineer wants to test the application, then he/she needs to access the test case only from the test case repository.
- If we do not require any test case, we can drop them from the test case repository.
- Each version have a separate test case repository.

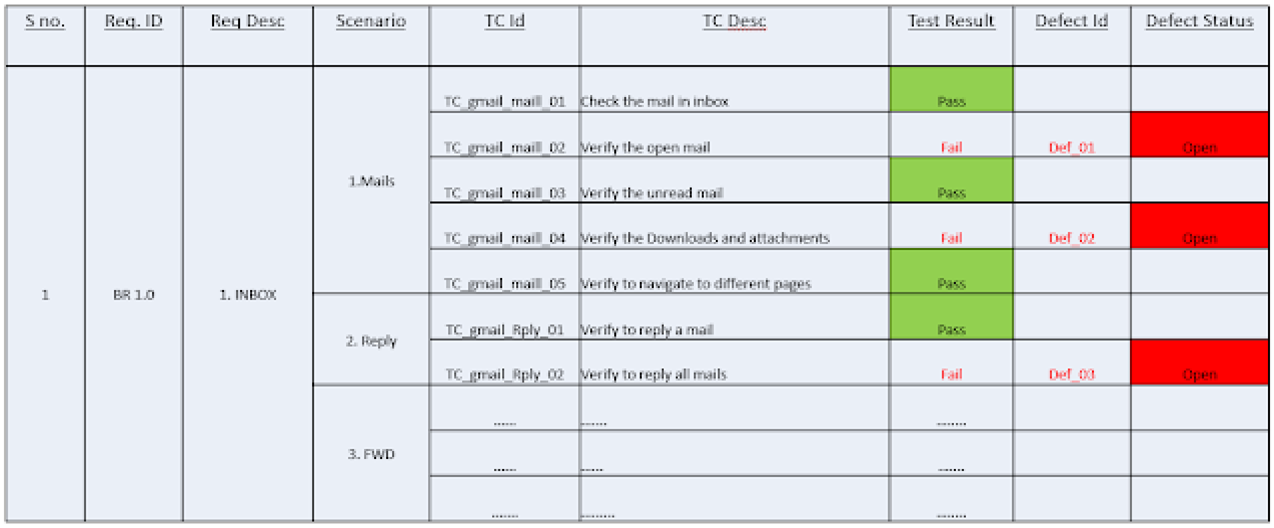
Requirement Traceability Matrix

Traceability Matrix:

- In this document the test cases are mapped to the corresponding requirement to ensure the coverage of test cases, to find any gap between requirement and test cases.
- It is also known as Requirement Traceability Matrix (RTM) or Cross Reference Matrix (CRM).
- It is prepared before the test execution process.
- This document is designed to make sure that each requirement has a test case.
- The test engineer will prepare TM for their respective assign modules, and then it will be sent to the Test Lead. Then test lead consolidate all and prepare one TM document for project.

Goals of Traceability Matrix:

- It allows you to see if a requirement is fully documented or not. A requirement traceability matrix can even call attention to missing requirements.
- It ensures that the software completely meets the customer's requirements.
- A traceability matrix can help in the effort to provide proper and consistent documentation for your team.
- It helps in detecting the root cause of any bug.



S no.	Req. ID	Req Desc	Scenario	TC Id	TC Desc	Test Result	Defect Id	Defect Status
1	BR 1.0	1. INBOX	1.Mails	TC_gmail_maill_01	Check the mail in inbox	Pass		
				TC_gmail_maill_02	Verify the open mail	Fail	Def_01	Open
				TC_gmail_maill_03	Verify the unread mail	Pass		
				TC_gmail_maill_04	Verify the Downloads and attachments	Fail	Def_02	Open
				TC_gmail_maill_05	Verify to navigate to different pages	Pass		
			2. Reply	TC_gmail_Rply_01	Verify to reply a mail	Pass		
				TC_gmail_Rply_02	Verify to reply all mails	Fail	Def_03	Open
			3. FWD	......	......	....		
				......	.....	......		
				....	....	....		

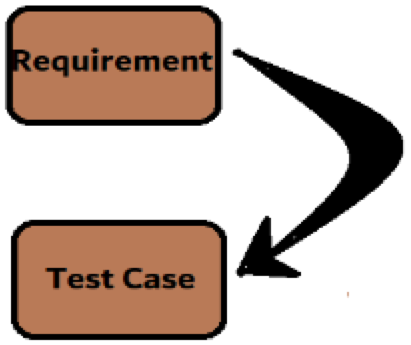
Types of Traceability Matrix:

The traceability matrix can be classified into three different types which are as follows:

- Forward Traceability
- Backward or reverse Traceability
- Bi-directional Traceability

Forward Traceability:

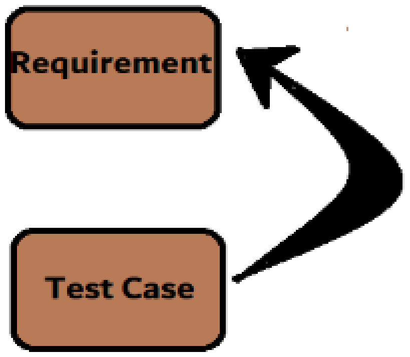
Forward traceability is used to map the requirements to the test cases.



- Not only this will establish that every requirement is being tested from top to bottom, but it will also assist in confirming that a project's trajectory is sound.
- The main objective of this is to verify whether the product developments are going in the right direction.

Backward or reverse Traceability:

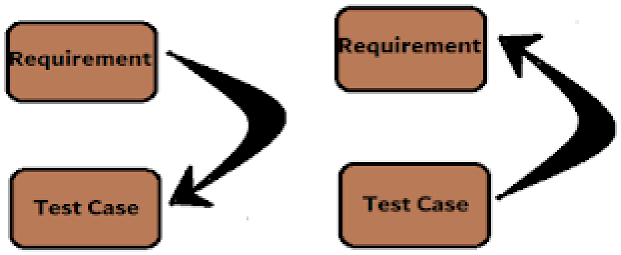
You can make a backward traceability matrix by mapping test cases with the requirements.



- The reverse or backward traceability is used to check that we are not increasing the space of the product by enhancing the design elements, code, test other things which are not mentioned in the business needs.
- The main objective of this that the existing project remains in the correct direction.

Bi-directional Traceability:

Bidirectional traceability essentially combines forward and backward traceability into one document.



- This type is useful because it establishes that each requirement has relating test cases.
- It also evaluates the modification in the requirement which is occurring due to the bugs in the application.

Test Execution Process | Environment setup

Test Execution:

- Test execution is the process of executing the test cases and comparing the expected and actual results.
- After developers finish the coding and give the build to the tester , tester will conduct the testing and validate the application as per the requirement.
- The developers will prepare the below document and give to the testing team during the build release :

SRN (software release note)
DD (Deployment Document)

- Tester deploy the build in test environment a per the instruction in the release document note .
- The build is deployed in the test environment either by tester or developer or n/w team.

- As we know after build release 1st test should be performed is sanity test .
- After Sanity test is pass we will continue test execution and validate all the functionality.
- Test execution can be done either manual aur automation.
- Testing performed by tester without using any tools is called manual testing.
- Testing conducted with the help of tool is called automation testing.
- During execution validate all the test cases and specify actual result and status.
- If the status is fail we can report that as defect to the developers.

SRN (software release note)

- A release note refers to the technical documentation produced and distributed alongside the launch of a new software product or a product update.
- It briefly describes a new product or specific detail level changes included in a product update.

- A software release notes template is a simple document that companies use to document any changes to their product. Elements of release notes template are like :
- Build version , build location , requirement in build , Precondition , environment , deployment steps , known issue , defect files , prepare by.

Defect Report | Priority and Severity

Defect Report:

- If application is not working as expected we can report that as defect in defect report.
- A defect report is a document that has concise details about what defects are identified, what action steps make the defects show up, and what are the expected results instead of the application showing error (defect) while taking particular step by step actions.
- Defect Reporting, Defect Tracking, and Status Tracking is called Defect Management. Some companies use Manual Process (Excel workbook), and some companies use Tool-based processes for Defect Management.
- Defect Management Tools Examples: Bugzilla / Issue-Tracker / PR-Tracker etc.

Important fields of a Defect Report template:

- Defect Id : It is a unique number for every defect we found while testing.
- Summary : Full description of the defect .
- Detected By : Name of the tester who found the defect .
- Assigned to : Currently the defect is assign to whom. When a tester found a defect the it has to assign to any developer.
- Severity : Severity defines how impactful a bug can be to the system. It can values either high or medium or low specifies the seriousness of the defect.
- Priority : Priority indicates how soon the bug should be fixed. It can values either high or medium or low specifies the importance of the defect or functionality. The defect are fixed based on priorities.

Defect Status in defect life cycle is the present state from which the defect or a bug is currently undergoing. The goal of defect status is to precisely convey the current state or progress of a defect in order to better track and understand the actual progress of the defect lifecycle.

- Build version : It show currently build version in which we are testing. We can check this from release note document.
- Module : In which module the defect is found.
- Environment : The defect is found in which environment like : Test environment, UAT or Production environment.
- Defect Type : Specify the type of defect like functional , network , data , performance or UI (User Interface) etc.
- Reproduceable : It has value either YES or NO:

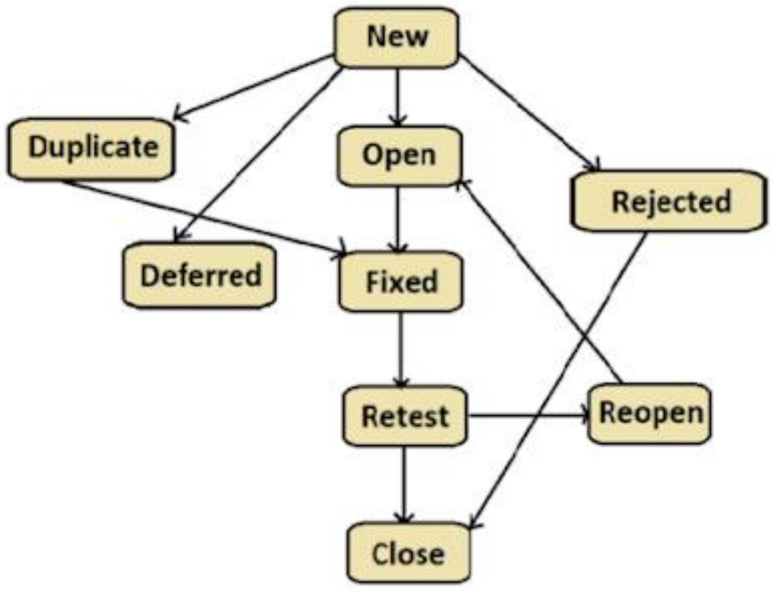
YES: Every time the defect is occurring .
NO: Most of the time its working correctly, only few cases it is failing and not able to reproduce

- (If the defect is non reproduceable we can take screen shot of the defect and report it along with the defect.)
- Reproduce Steps : Specify the navigation steps to reproduce the defect.

Defect/Bug Life Cycle

Defect Life Cycle:

- Defect life cycle, also known as Bug Life cycle is the journey of a defect cycle, which a defect goes through during its lifetime.
- It varies from organization to organization and also from project to project as it is governed by the software testing process and also depends upon the tools used.
- Defect Life Cycle is a cyclic process, which describes how a defect or bug passes through different stages from the identification stage to the Fixing stage. it begins when a tester finds or logs a bug and it ends when the bug is fixed.
- If Developer reject a defect understand the reason why defect is rejected , if developer is correct we can close the defect or else we can discuss it with test lead or project lead and than reopen the defect.



Some of the familiar values used for defects, during their life cycle are :

- New : New defect is reported.
- Open : The developer is currently working on the defect.
- Fixed : The code changes are completed and the defect is resolved .
- Deferred : Developers will fixed the defect later.

- Duplicate : The defect is same like one of the previous defect. When previous is fixed it also fixed.
- Retest : the tester starts the task of retesting the defect to verify if the defect is fixed or not.
- Rejected : Developers did not accept the defect .
- Close : After re test if defect is working correctly .
- Reopen : After re test if defect is still exist then we reopen the defect.
- Not a Bug : If the defect does not have an impact on the functionality of the application, then the status of the defect gets changed to “Not a Bug”.
- There are others of course, and some groups might use combinations of these values (such as Closed-Fixed, Closed-WontFix, etc.).

Types of Software Testing

In this post, we will describe different types of software testing. Various types of software testing are performed to achieve different objectives when testing a software application.

Unit Testing or Component Testing: UNIT TESTING is a level of software testing where individual units/ components of a software are tested. ... A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc.This testing is conducted by developer or white box tester.
For Example : if a developer is developing a loop for searching functionality of an application which is a very small unit of the whole code of that application then to verify that the particular loop is working properly or not is known as unit testing.

Structural Testing Techniques: Structural testing is the type of testing carried out to test the structure of code. It is also known as White Box testing or Glass Box testing. This type of testing requires knowledge of the code, so, it is mostly done by the developers.
For Example : Testing each line in the programme at least one time , verify the conditional statement are working properly or not

Different method of Structural Testing Techniques

Branch testing: Branch Testing is defined as a testing method, which has the main goal to ensure that each one of the possible branches from each decision point is executed at least once and thereby ensuring that all reachable code is executed.
Path Testing: Path testing is a structural testing method that involves using the source code of a program in order to find every possible executable path. It helps to determine all faults lying within a piece of code. This method is designed to execute all or selected path through a computer program
Statement testing: A test strategy in which each statement of a program is executed at least once. It is equivalent to finding a path (or set of paths) through the control-flow graph that contains all the nodes of the graph.
Condition Testing: It is another structural testing method that is useful during unit testing, using source code or detailed pseudocode as a reference for test design. Its goal

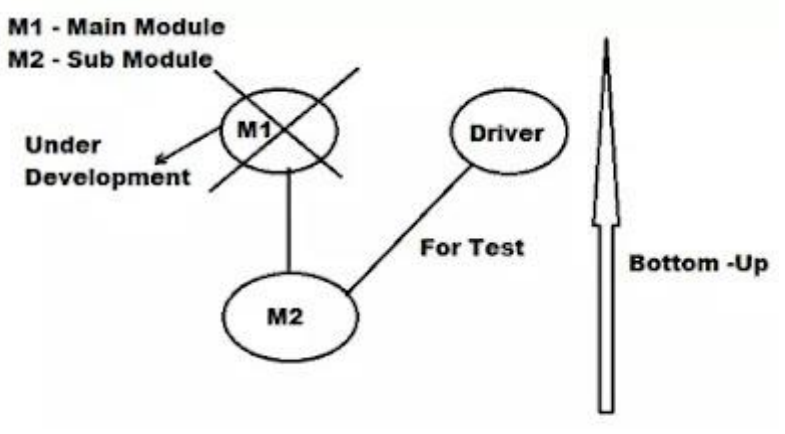
is the thorough testing of every condition or test that occurs in the source code.

Functional Testing: FUNCTIONAL TESTING is a type of software testing whereby the system is tested against the functional requirements/specifications. Functions (or features) are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application. Functional testing is a quality assurance process and a type of black-box testing that bases its test cases on the specifications of the software component under test
Examples of Functional testing are. Unit Testing, Smoke Testing, Sanity Testing, Integration Testing etc.

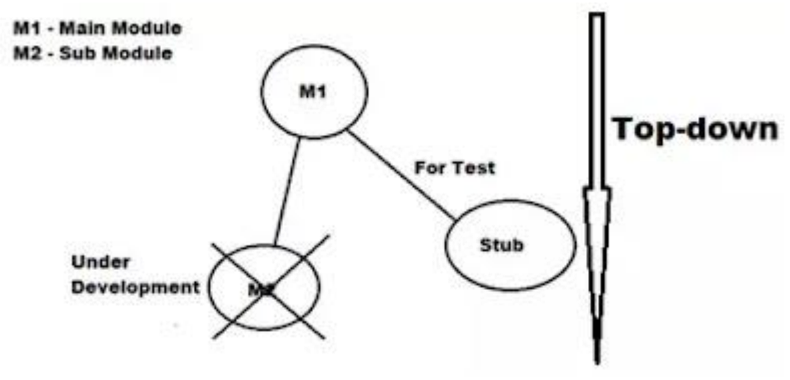
Error Based Testing: Testing is conducted to analyze the impact of the defect on the application. Intentionally adding the defect in application.

Mutation Testing: Mutation Testing is a type of software testing where we mutate (change) certain statements in the source code and check if the test cases are able to find the errors. It is a type of White Box Testing which is mainly used for Unit Testing.

Integration testing: INTEGRATION TESTING is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and Test stubs are used to assist in Integration Testing.
Test Driver and Stub: This are the temporary components which are used in place of main and sub module
Driver: A substitute for main module in bottom-up integration



Bottom-up Integration
Stub: It is a substitute for sub module in Top-down integration



Top-Down Integration

Different type of approaches in integration testing

1. Big Bang integration: Big Bang Integration Testing is the Integration testing where no incremental testing takes place prior to all the system's components being combined to form the system. In this approach, individual modules are integrated until all the modules are ready, and all or most units are combined and tested in one go. However, if there are too many modules and the system is multilevel and complex, it is better not to apply the big-bang integration testing.

2. Top-down integration: Top-down testing is an approach to integrated testing where the top integrated modules are tested and the branch of the module is tested step by step until the end of the related module.

3. Bottom – up Integration: A type of integration testing, bottom-up approach is a testing strategy in which the modules at the lower level are tested with higher modules until all the modules and aspects of the software are tested properly.

4. Mixed / Sandwich integration: Sandwich Testing is the combination of bottom-up approach and top-down approach, so it uses the advantage of both bottom up approach and top down approach. ... It is also known as the Hybrid Integration Testing

System Testing: SYSTEM TESTING is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. System testing takes, as its input, all of the integrated components that have passed integration testing. It's a black box testing . during system test we will conduct both functional and non functional testing.

User Interface Testing: Verify the page or window displayed , testing technique used to identify the presence of defects is a product/software under test by using Graphical user interface [GUI].

Object Properties: Verify the properties of an object like enabled , disabled ,colors, X and Y coordinates etc .

Error guessing: Testing is conducted by performing invalid operations to validate the error msg is displaying or not.

Input Domain Testing: Validate the data accepted by the system , the testing is conducted for the Edit box, input fields etc.

Database Testing: Database Testing is a type of software testing that checks the schema, tables, triggers etc. of the database under test. It involves creating complex queries for performing the load or stress test on the database and check its responsiveness. It checks integrity and consistency of data.

- Validate the data in table
- Constrain primary, foreign key
- Data migration
- Security
- Performance

Calculation Testing: The test is important for bank application , functional application etc. to verify all the calculated values are correct.

Links and URL's Testing: this test is only for web application , verify all the links are working correct or not, verify the navigation is as per expected or not. Verify the text link , image link , broken links etc.

Non functional Testing: NON-FUNCTIONAL TESTING is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.

Usability testing: Usability testing is a technique used in user-centered interaction design to evaluate a product by testing it on users. This can be seen as an irreplaceable usability practice, since it gives direct input on how real users use the system.

Load Testing: Load testing is a type of non-functional testing. A load test is type of software testing which is conducted to understand the behavior of the application under a specific expected load. Load testing is performed to determine a system's behavior under both normal and at peak conditions.

Stress Testing: STRESS TESTING is a type of Software Testing that verifies the stability & reliability of the system. This test mainly measures the system on its robustness and error handling capabilities under extremely heavy load conditions. Stress Testing is done to make sure that the system would not crash under crunch situations.

Soak Testing: Soak testing involves testing a system with a typical production load, over a continuous availability period, to validate system behavior under production use. It may be required to extrapolate the results, if not possible to conduct such an extended test.
Note Conducting Load , stress testing etc… for 100 of users is not possible to do manually the tools are :

- Load Runner
- Jmeter Web load
- RPT (rational performance test tool)
- Silk Performance

Memory Testing/ Memory leakage testing: Some memory got allocated and not used for any purpose are called memory leakage. When accessing the application verify the usage of memory. To find a memory leak, you've got to look at the system's RAM usage.

Volume Testing: VOLUME TESTING is a type of Software Testing, where the software is subjected to a huge volume of data. It is also referred to as flood testing. Volume testing is done to analyze the system performance by increasing the volume of data in the database. Usually conducted when accessing the data from database , performance is calculated by increasing the data transfer b/w the system.

Performance test: Performance testing is the process of determining the speed, responsiveness and stability of a computer, network, software program or device under a workload. Performance testing can involve quantitative tests done in a lab, or occur in the production environment in limited scenarios. This test is needed for application which are accessed by more no of users, accessed from remote locations.
During performance test we are calculate:

- **Response time:** Time taken to get response from server to client.
- **Hit per second:** No of request revived by the server in 1 second of time
- **Throughput**: Throughput – indicates the number of transactions per second an application can handle, the amount of transactions produced over time during a test.
- **Elapsed Time:** Elapsed time is measured by the time from the first moment of sending the data and the time of the last byte of the received response.

Performance of application depends on:

- Configuration of the system
- Network Speed
- Load on the server
- Configuration Setting
- Programming Logic

Agile Methodology

What is Agile model?

It is a process/model of continuous integration of development and testing throughout the software development life cycle of the project. It is one of the most widely used process in software development world as it helps teams to deliver the work faster in small increments with less issues.

Agile Process:

Here in Agile process, we can monitor the project progress every day. Every release is implemented in iterations or sprints. Every iteration or sprint consists of 2 or 3 weeks in general. However, this sprint duration varies from project to project and also to sprint to sprint as well sometimes. Many meetings were involved in this Agile process to know the progress of the release in each sprint. However below mentioned meetings are widely conducted. Let's see about them one by one in detail,

Ø **Sprint Planning meeting** is conducted before the start of the sprint to discuss the user stories that can be covered in that particular sprint.

Ø **Daily stand up/Scrum** is conducted to know the work done by each in the team. Mainly the below things are discussed,

- What work is done yesterday
- What work is planned for today
- What work is planned for tomorrow
- Any impediments to complete the work

Ø **Sprint Review meeting** is conducted at the end of the sprint to discuss the progress and where does the team stand for that sprint. However this is a optional meeting which is conducted based on the projects/companies as required

Ø **Sprint Retrospective meeting** is conducted once a sprint is completed and next sprint got started. Following things will be discussed,

- What went well in the last sprint
- What didn't went well in the last sprint
- What can be done to improve or perform well in the ongoing sprint

Ø **Defect Triage meeting** is conducted to review and have a proper action on the defects. Dev team, testing team and sometimes the Product owner also will be involved in this meeting. Defects identified in the particular sprint will be triaged and assigned to respective team or deferred sometimes depending on the defects.

Ø **Sprint Planning meeting** is conducted before the start of the sprint where the user stories for that particular sprint and the timelines for the same would be planned.

General terminologies used in Agile process

Product Backlog – User stories(requirements are split into user stories) will be added here. For any sprint, user stories will be picked from product backlog and worked upon.

Epic – It is the complete feature or requirement which needs to be implemented.

User story – An Epic is split into user stories based on the complexity or work involved in order to develop it. User story is generally part of Epic. In case of small feature, it will be a single user story.

For easy understanding of difference between epic and user story, let's look at a simple example. Consider if we need to implement a login page functionality, it can be treated as an Epic. "Username, password" fields functionality can be considered as one user story. Any other hyperlinks such as "Forgot password, Need help" can be considered as another user story.

Story points – It is a measurement to define the complexity of the user story from dev and QA perspective. This activity will be done in Sprint planning.

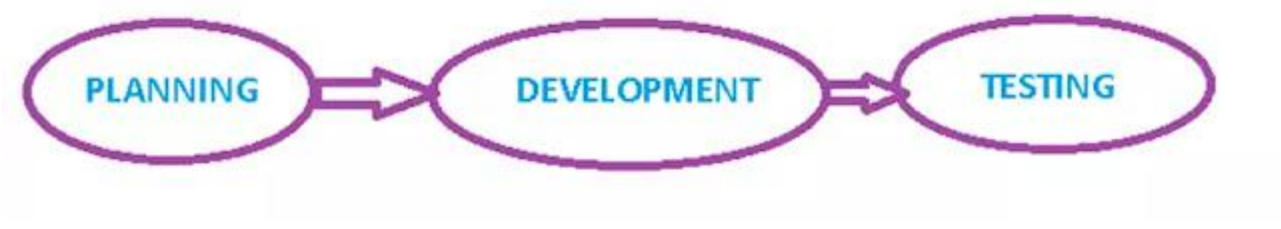
Burn down chart – This is a pictorial graph representation of time and work. This is drawn with the help of the time spent on tasks along with the work. Team need to burn the hours on daily basis for the work accomplished against the tasks.

Differences between Waterfall and Agile

Let's see what are the basic differences between Waterfall and Agile models.

Waterfall is a traditional model which is also called as Linear Sequential model.

- This is mainly suitable for small projects or where the requirements are planned prior/freeze.
- In this model, each phase is done only after previous phase/step is successfully completed.
- Difficult to initiate change in the requirement once the project is started.
- Testing is performed only after the completion of development.
- Each phase has specific deliverable and review process.
- Documentation is maintained well in each phase and dependencies can be easily addressed.



Agile model is where continuous development and testing is done. Communication is more or we can say involvement of all teams happens in this model.

- It is an incremental process.
- This process is known for its flexibility.
- Changes in the project can be done even if the initial plan is completed.

- Testing is performed concurrently with software development.
- Less dependency on the project manager and team member can work on multiple projects in the same moment.
- Client is involved in every phase and easy to know the updates of the project.

What is Scaled Agile Framework

It is also known as SAFe. This methodology is an agile framework for development teams. The framework is divided into three segments: **Team, Program** and **Portfolio.**

Following are allowed in SAFe,

- To implement Lean-Agile software and systems in enterprise level
- Gives detailed guidance for work at enterprise
- Designed based on Lean and Agile principles
- Designed to meet the needs of organization

Benefits of Scaled Agile Framework

It is lighter in weight and simpler. Also it can be used to handle large size streams. Few of the key benefits are listed below,

- Increases the productivity
- Helps in increasing the employee engagement and job satisfaction
- Decreases time taken for the **time to market**
- Increases the quality

How is this different from other Agile practices

- This is free to use
- It is lightweight and publicly available to use
- Maintains most commonly used agile practices
- Have transparency on all the levels
- Available on easy and highly approachable form

Best Agile and Scrum Certifications

Following are the best certifications which can help the individuals in their careers,

- Certified Scrum Professional
- Certified Scrum Master
- Agile Certified Coach
- Agile Certified Practitioner
- SAFe 4 Agilest

Best Agile Project Management Tools

Following are the most commonly used project management tools,

- Zephyr
- Atalssian JIRA
- SoapUI
- Kanbanize
- VSTS
- Pivotal Tracker

These are some of the basic things about Agile methodology.

What is Codeless Automated Testing?

Codeless automated testing *is a testing approach that allows testers to create and execute automated test cases without the need to write code. Instead of writing code, testers use a visual interface or a set of pre-defined commands to create and execute test cases.*

One of the main benefits of codeless automated testing is that it allows testers with limited coding skills to create and execute automated tests. This can help to accelerate the testing process and make it more accessible to a wider range of testers. For example, a tester who is not familiar with programming languages can still create and run automated tests using a codeless testing tool.

Another benefit of codeless automated testing is that it can reduce the time and effort required to maintain test cases. Since no code needs to be written, there is less need to update and maintain test cases as the software changes. This can help to reduce the overall cost of testing and make it more efficient.

There are several different tools and platforms available for codeless automated testing. These tools typically provide a visual interface or a set of pre-defined commands that testers can use to create and execute test cases. Some common features of codeless automated testing tools include:

Record and playback functionality: This allows testers to record their actions as they test the software manually, and then play them back automatically to create an automated test case.

Object recognition: This allows the tool to identify and interact with specific elements on the screen, such as buttons or input fields.

Test case management: This allows testers to organize and manage their test cases, including the ability to create and edit test cases, view test results, and track defects.

Test data management: This allows testers to create and manage test data, including the ability to create and edit test data, view test results, and track defects.

Integration with other tools: Many codeless automated testing tools can be integrated with other tools and platforms, such as defect tracking systems, continuous integration systems, and test management systems. This can help to streamline the testing process and make it more efficient.

While codeless automated testing can offer many benefits, it is important to note that it may not be suitable for all testing scenarios. In some cases, writing custom code may be necessary to fully test the software. For example, custom code may be required to test complex logic or to interact with third-party systems or APIs. Additionally, codeless automated testing may not provide the same level *of flexibility and control as code-based automated testing.*

Despite these limitations, codeless automated testing can be a useful approach for testers who want to create and execute automated tests without the need to write code. It can help to accelerate the testing process and make it more accessible to a wider range of testers, while also reducing the time and effort required to maintain test cases.

In conclusion, codeless automated testing is a valuable tool for testers who want to create and execute automated tests without the need to write code. It can help to accelerate the testing process and make it more efficient, while also making it more accessible to a wider range of testers. However, it is important to carefully evaluate the suitability of codeless automated testing for a given testing scenario, as it may not always be the best fit.

There are several codeless automation tools available in the market, each with its own unique features and capabilities. Some of the more popular codeless automation tools include:

***Selenium IDE**: This is an open-source browser extension that allows testers to record and play back test cases in the browser. It supports a wide range of browsers, including Chrome, Firefox, and Safari.*

***TestComplete**: This is a commercial automated testing tool that allows testers to create and execute test cases for desktop, web, and mobile applications. It features an easy-to-use visual interface and supports a wide range of programming languages.*

***TestProject**: This is an open-source automated testing platform that allows testers to create and execute test cases for web, mobile, and API applications. It features a visual interface and integrations with popular defect tracking and continuous integration tools.*

***UiPath**: This is a commercial robotic process automation (RPA) platform that allows users to automate repetitive tasks by creating and executing automated workflows. It features a visual interface and integrations with a wide range of applications and systems.*

***Katalon Studio**: This is a free, open-source automated testing platform that allows testers to create and execute test cases for web, mobile, and API applications. It features a visual interface and supports a wide range of programming languages.*

***Testim**: This is a commercial automated testing platform that allows testers to create and execute test cases for web and mobile applications. It features a visual interface and integrations with popular defect tracking and continuous integration tools.*

These are just a few examples of the many codeless automation tools available in the market. It is important to carefully evaluate the features and capabilities of each tool to determine which one is the best fit for your specific testing needs.

Selenium Cucumber Java BDD Framework:

Intoduction to Behaviour Driven Development BDD Framework

In this post we will discuss about the following points:

- Traditional Approach Vs TDD
- What is BDD?
- TDD Vs BDD
- Agile and BDD
- Cucumber, Gherkin and BDD Example
- BDD Tools

Traditional Approach

A traditional development process would go as follows:

1. Business needs - BRD.
2. Product owners write requirements based on those business needs - SRS, FRS.
3. Developers and testers (independently) translate those requirements into code and test cases.

So Traditional Approach (Driven by coding) would be:

Coding -> Test Case Creation -> Test Execution

BRD (Business requirement Document)- Initiation phase- Why the requirements are being undertaken.

SRS (Software Requirement Specification)- Planning Phase -What requirements must be fulfilled to satisfy business needs.

FRS (Functional Requirement Specification)- Planning phase - How exactly the system is expected to function.

TDD (Test Driven Development)

1. Business needs.
2. Create a collaborative space with developers and testers so that they can talk about what they’re going to develop together.
3. During this phase they collaborate around requirements and define them as English format scenarios.

4. Developers write code based on these scenarios and testers create automated test cases that report back against these scenarios.

So Test Driven Development would be:

Tests are created -> Run the tests (Will fail) -> Write the code to pass the tests

What is BDD?

- Behavioral Driven Development (BDD) is a software development approach that has evolved from TDD (Test Driven Development).
- Behavior Driven Development (BDD) is a branch of Test Driven Development (TDD).
- It Improves communication between tech and non-tech teams and stakeholders.
- In both development approaches, tests are written ahead of the code, but in BDD, tests are more user-focused and based on the system's behavior.
- BDD is a way for software teams to work that closes the gap between business people and technical people.
- BDD is more user focused – for this BDD use user stories for every feature to be tested Ex – Login feature.
- BDD uses human readable descriptions of user requirements – Writing the scripts using Gherkin Language.

TDD vs. BDD

- BDD is in a more readable format by every stake holder since it is in English, unlike TDD test cases written in programming languages such as Ruby, Java etc.
- BDD explains the behavior of an application for the end user while TDD focuses on how functionality is implemented. Changes on functionality can be accommodated with less impact in BDD as opposed to TDD.

Agile and BDD

- Agile Planning work in small increments of value like User Stories.
- BDD does not replace your existing agile process, it enhances it.
- BDD as a set of plug-in for your existing process that will make your team more able to deliver on the promises of agile:
- BDD encourages working in rapid iterations, continuously breaking down your user's problems into small pieces that can flow through your development process as quickly as possible.

What is Cucumber:
It is a testing tool that supports Behavior Driven Development (BDD) framework. It defines application behavior using simple English text, defined by a language called Gherkin.

Gherkin Language:

It is a Business Readable, Domain Specific Language created especially for behavior descriptions.

The ‘Given-When-And-Then’ formula BDD example
This is the proposed template for writing BDD test cases for a user story, which can be defined as:

Feature: Description of the feature
Scenario:
Given a certain scenario which is prerequisite
When an action takes place - Action
And some more action
Then this should be the outcome – Expected Result

A practical example would be:-

Feature: Check Login Functionality
Scenario:
Given the login screen is opened
When user enters username and password
And clicks on Login Button
Then user should be navigated to Home Page.

Different BDD Tools:

Cucumber:
Cucumber is a test framework that supports BDD. In Cucumber, the BDD specifications are written in plain, simple English which is defined by the Gherkin language. In other words, Gherkin is a language that Cucumber understands.
Specflow:
Specflow evolved from the Cucumber framework using Ruby on Rails, and is used mainly for .Net projects. SpecFlow also uses the Gherkin language.
Jbehave
Test Left
JDave

Behaviour Driven Development (BDD) Framework Setup

Steps to be followed to setup BDD Framework:

1. Create a Maven Project

2. Add dependencies - Selenium java, webdrivermanager, cucumber-java, junit, cucumber-junit

3. Create Features folder in src/test/resources

4. Create a new feature file. extension is .feature

5. Install Cucumber Eclipse plug-in or natural plug-in and Restart the eclipse.

6. Create a new feature file to get all the options

7. Write Scenarios in feature file

8. Run the feature file

9. Create step definition class or glue code under src/test/java package

10. Create a runner package/class

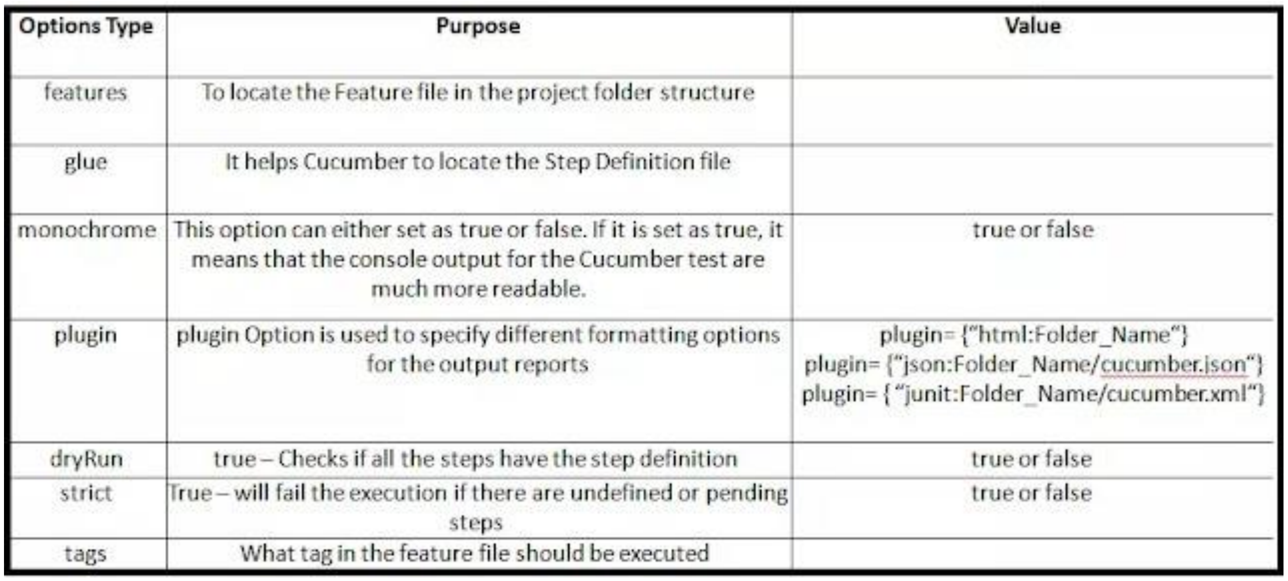
@RunWith(Cucumber.class)

@CucumberOptions(features="src/test/resources/Features",glue={"com.Step Definition"}

Please refer below YouTube video:

What is Cucumber Options?
@CucumberOptions are like property files or settings for your test. Basically @CucumberOptions enables us to do all the things that we could have done if we have used cucumber command line.

Different Cucumber Options:



Options Type	Purpose	Value
features	To locate the Feature file in the project folder structure	
glue	It helps Cucumber to locate the Step Definition file	
monochrome	This option can either set as true or false. If it is set as true, it means that the console output for the Cucumber test are much more readable.	true or false

plugin	plugin Option is used to specify different formatting options for the output reports	plugin= {"html:Folder_Name"}
		plugin= {"json:Folder_Name/cucumber.json"}
		plugin= { "junit:Folder_Name/cucumber.xml"}

dryRun	true – Checks if all the steps have the step definition	true or false
strict	True – will fail the execution if there are undefined or pending steps	true or false
tags	What tag in the feature file should be executed	

Data Driven Testing in Cucumber

What is Cucumber data driven testing?

In very simple terms, cucumber data driven testing means that you can pass data from cucumber feature files to your test cases. For example, let us consider that you are automating the login flow of some application. In this case, you can pass user name and password as test data from your feature file to selenium code.

Data-Driven Testing in Cucumber

1. Parameterization without Example Keyword:
Parameterization without Example Keyword
Using Regular Expression:
("^user enters \"(.*)\" and \"(.*)\"$")
("^user enters \"([^\"]*)\" and \"([^\"]*)\"$")

2. Data-Driven Testing in Cucumber using Scenario Outline
Parameterization with Example Keyword

Parameterization with Examples Keyword
Using Regular Expression:
("user enters (.*) and (.*)$")

Scenario Outline – This is used to run the same scenario for 2 or more different sets of test data. E.g. In our scenario, if you want to register another user you can data drive the same scenario twice.

Examples – All scenario outlines have to be followed with the Examples section. This contains the data that has to be passed on to the scenario.

Using Scenario Outline

Cucumber inherently supports data driven testing using Scenario Outline. Consider the following feature file using Scenario to define the test steps-

```
 Feature: Check addition in Google calculator
   In order to verify that google calculator work correctly
   As a user of google
   I should be able to get correct addition result

   Scenario: Addition
   Given I open google
   When I enter "2+2" in search textbox
   Then I should get result as "4"
```

In order to make it data driven we just have to use Scenario Outline along with Examples. Write the following code in feature file-

```
   Feature: Check addition in Google calculator
   In order to verify that google calculator work correctly
   As a user of google
   I should be able to get correct addition result

   Scenario Outline: Addition
   Given I open google
   When I enter "<calculation>" in search textbox
   Then I should get result as "<result>"

   Examples:
        | calculation |result|
        | 3+3         | 6  |
        | 2+5         | 9  |
```

What is Data table in Cucumber?

Data Tables are handy for passing a list of values to a step definition. Cucumber provides a rich API for manipulating tables from within step definitions.
Data Tables are also used to handle large amount of data. Tables in Cucumber feature files are represented by using the pipeline "|" sign.

Database and SQL Concept:

SQL Part 1: DataBase Concept

In This section we are going to cover below topics:

- SQL For Testers
- What is Database?
- What is DBMS?
- What is RDBMS?
- What is SQL?
- SQL Command Types
- Most Important SQL Commands

Why SQL needed for Testers

- The ability to recognize the different types of databases, The ability to connect to the database using different SQL connection clients, Understanding of the relationship between database tables, keys, and indices. Ability to write a simple select or SQL statement along with more complex join queries.
- As a software tester, you are required to perform database testing that requires the knowledge of different SQL and database concepts. In addition, you are required to write SQL queries to retrieve, update and insert data in the databases.
- We will start with the database fundamentals. At the same time, we will move to the SQL concepts. Finally, we'll check some of the widely used SQL commands

What is a Database?

- A database is an organized and systematic collection of data that is stored and accessed in/from a computer system. A database is usually controlled by a database management system (DBMS).
- For example, a company database may include tables for products, employees, and financial records. Each of these tables would have different fields that are relevant to the information stored in the table.

What is DBMS?

- DBMS (Database Management System) is a software system that is designed to maintain and access the database. It allows the user to access, create, delete, and update data in a database.
- DBMS defines the rules for manipulation and validation of this data. We use a database when there is a huge amount of data, the security of the data is important, or when multiple users have to have access to the data concurrently.
- Some DBMS examples include MySQL, PostgreSQL, Microsoft Access, SQL Server, FileMaker, Oracle, RDBMS, dBASE, Clipper, and FoxPro

What is RDBMS?
RDBMS (Relational Database Management System) is an advanced version of the basic DBMS.
A relational database is a database that allows the user to store related data in the form of multiple tables, which are linked by establishing a relationship between the different tables, hence providing an efficient way to access the database.

- Top Relational Databases Software
- MS SQL
- Oracle Database
- MySQL
- Db2
- PostgreSQL

- A RDBMS database uses tables for storing the data. A table is nothing but a collection of data that is related to one another.
- **Rows**
- A Row represents a collection of fields that end up making a record in the database.
- Column
- In a database, a column represents those values that are of the same type. A column is also called an attribute.

What is SQL?

- SQL stands for Structured Query Language. It is a programming language that is used to request information from a database. SQL can be used to manage and share data in a relational database management system. Moreover, users can perform actions like insertion, deletion, selection, etc on the database.

- SQL keywords are not case sensitive.

SQL commands are segregated into following types:

- DDL – Data Definition Language –

SQL commands come under DDL are as follows: CREATE, ALTER, DROP, TRUNCATE

- DML – Data Manipulation Language

SQL Commands come under DML are as follows: INSERT, UPDATE, DELETE

- DQL – Data Query Language

SQL Commands come under DQL are as follows: SELECT

- DCL – Data Control Language

SQL Commands come under DCL are as follows: GRANT, REVOKE, DENY

- TCL – Transaction Control Language

SQL Commands come under TCL are as follows: COMMIT, ROLEBACK, SAVE

Some frequently used keywords/commands:

- SELECT – It extracts data from a DataBase
- INSERT INTO – It inserts new data into a DataBase
- CREATE – It creates a new DataBase/Table/Index
- UPDATE – It updates data in a DataBase
- ALTER – It modifies a DataBase/Table

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.
The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

Syntax: ALTER TABLE table_name
ADD column_name datatype;

- TRUNCATE - Truncate can be used to delete the entire data of the table. Table structure remain same.

Syntax: TRUNCATE TABLE table_name

- DELETE – Delete statement can be used for deleting the specific data.

Syntax/Example:
Delete from <table>;
Delete from <table> where ID=3;

- DROP – It deletes a table/Index.

Syntax: DROP TABLE table_name

SQL Part 2: Data types, Operators and Constraints

SQL Datatypes:

Exact Numeric SQL Data Types:

bigint = Range from -2^63 (-9,223,372,036,854,775,808) to 2^63-1 (9,223,372,036,854,775,807)

int = Range from -2^31 (-2,147,483,648) to 2^31-1 (2,147,483,647)

smallint = Range from -2^15 (-32,768) to 2^15-1 (32,767)

tinyint = Range from 0 to 255

bit = 0 and 1

decimal = Range from –10^38 +1 to 10^38 -1

numeric = Range from -10^38 +1 to 10^38 -1

money = Range from -922,337,203,685,477.5808 to +922,337,203,685,477.5807

small money = Range from -214,748.3648 to +214,748.3647

Approximate Numeric SQL Data Types:

float = Range from -1.79E + 308 to 1.79E + 308

real = Range from -3.40E + 38 to 3.40E + 38

Date and Time SQL Data Types:

datetime = From Jan 1, 1753 to Dec 31, 9999

smalldatetime = From Jan 1, 1900 to Jun 6, 2079

date = To store a date like March 27, 1986

time = To store a time of day like 12:00 A.M.

<u>Character Strings SQL Data Types:</u>

char = CHAR is fixed lengt. Maximum length of 8,000 characters

varchar = VARCHAR is variable length. Maximum of 8,000 characters

varchar(max) = Maximum length of 231 characters

<u>char vs varchar</u>

CHAR takes up 1 byte per character. So, a CHAR(100) field (or variable) takes up 100 bytes on disk, regardless of the string it holds. VARCHAR is a variable length string data type, so it holds only the characters you assign to it. VARCHAR takes up 1 byte per character, + 2 bytes to hold length information.

<u>Unicode Character Strings SQL Data Types:</u>

nchar = Maximum length of 4,000 characters

nvarchar = Maximum length of 4,000 characters

nvarchar(max) = Maximum length of 231 characters

ntext = Maximum length of 1,073,741,823 characters

<u>Binary SQL Data Types:</u>

binary = Maximum length of 8,000 bytes

varbinary ☐ = Maximum length of 8,000 bytes

varbinary(max) = Maximum length of 231 bytes

image = Maximum length of 2,147,483,647 bytes

SQL Operators:

There are are three types of Operators.

- Arithmetic Operators:
- Comparison Operators
- Logical Operators

Arithmetic operators

Operator	Description
+	Add values of operands
–	Subtract values of operands
*	Multiply operand's values
/	Divide values of operands
%	Modulus operation on operands

Comparison operators

In the table below, if condition gets satisfied then "True" Boolean value is returned.

Operator	Description
=	Determine if the values of operands are equal.
!=	Check if the values of operands are not equal.
>	Determine if the left operand is more than the right operand.
<	Check if the right operand is more than the left operand.
>=	Determine if the left operand is more than or equal to the right operand.
<=	Check if the right operand is more than or equal to the left operand.
!>	Determine if the right operand is not more than the left operand.
!<	Check if the left operand is not more than the right operand.

Logical Operators

Operator Description

OR - Returns true if either operand is true. Else it returns false if both the operands are false.

AND - Returns true if both operands are true. Else it returns false if either or both the operands are false

NOT - Returns true if condition is false and returns false if the condition is true.

SQL Constraints:

- Constraints are the rules enforced on data columns on table.
- These are used to limit the type of data that can go into a table.
- This ensures the accuracy and reliability of the data in the database.
- Constraints could be column level or table level.
- Column level constraints are applied only to one column,
- whereas table level constraints are applied to the whole table.

Following are commonly used constraints available in SQL:

1. NOT NULL Constraint: Ensures that a column cannot have NULL value.

Example:

```
CREATE TABLE recipes (

  recipe_id INT NOT NULL,

  recipe_name VARCHAR(30) NOT NULL,

  PRIMARY KEY (recipe_id),

  UNIQUE (recipe_name)

);
```

2. DEFAULT Constraint: Provides a default value for a column when none is specified.

3. UNIQUE Constraint: Ensures that all values in a column are different.

4. PRIMARY Key: Uniquely identified each rows/records in a database table.

UNIQUE KEY vs PRIMARY KEY

- The UNIQUE constraint ensures that all values in a column are different.
- Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint.
- However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.
- The primary key column cannot have null values while the Unique Key column can have one null value.

5. FOREIGN Key: Uniquely identified a rows/records in any another database table.

- A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table. The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.
- Foreign key is used to maintain relationship between two tables.
- Foreign Key in a table doesn't have to be unique in said table. BUT, as it is the Primary Key of another table, it must be unique in this table.
- Foreign Key can be NULL or duplicate

Example:

Table1 (Parent Table):
CREATE TABLE EmployeeDetail(EmpId integer PRIMARY KEY, FirstName varchar(255),LastName varchar(255),Salary integer,JoiningDate date, Department varchar(255), Gender varchar(255));

Table 2 (Child Table):

CREATE TABLE ProjectDetail(ProjectDetailId integer PRIMARY KEY, EmpId integer REFERENCES EmployeeDetail(EmpId), ProjectName varchar(255));

6. CHECK Constraint: The CHECK constraint ensures that all values in a column satisfy certain conditions.

SQL constraints are used to specify rules for the data in a table. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Syntax: ALTER TABLE table_name ADD CONSTRAINT constraint_name CHECK (column_name condition);

7. INDEX: Use to create and retrieve data from the database very quickly.

A SQL index is used to retrieve data from a database very fast. Indexing a table or view is, without a doubt, one of the best ways to improve the performance of queries and applications.

Syntax: CREATE INDEX <indexName>

ON <TableName>(LastName);

SQL Part 3: Create new table, Alias, Merge columns

<u>Query to Create EmployeeDetail table:</u>

```
CREATE TABLE EmployeeDetail

(

EmpId integer PRIMARY KEY,

FirstName varchar(255),

LastName varchar(255),

Salary integer,

JoiningDate date,

Department varchar(255),

Gender varchar(255)

);

INSERT INTO EmployeeDetail VALUES(1,'Tom','Garcia',60000.00,'2008-11-10
13:23:44','IT','Male');

INSERT INTO EmployeeDetail VALUES(2,'Lucy','Martin',70000.00,'2009-11-09
13:23:44','HR','FeMale');

INSERT INTO EmployeeDetail VALUES(3,'Frank','Clark',75000.00,'2012-01-24
13:23:44','Admin','Male');

INSERT INTO EmployeeDetail VALUES(4,'Jane','Joseph',78000.00,'2017-01-28
13:23:44','ICT','FeMale');

INSERT INTO EmployeeDetail VALUES(5,'Robert','Hansen',100000.00,'2012-01-29
13:23:44','HR','Male');
```

<u>Query to Create ProjectDetail table:</u>

```
CREATE TABLE ProjectDetail

(

ProjectDetailId integer PRIMARY KEY,
```

EmpId integer REFERENCES EmployeeDetail(EmpId),

ProjectName varchar(255)

);

INSERT INTO ProjectDetail VALUES(1,1,'ABC');

INSERT INTO ProjectDetail VALUES(2,1,'DDC');

INSERT INTO ProjectDetail VALUES(3,1,'GHI');

INSERT INTO ProjectDetail VALUES(4,2,'HGT');

INSERT INTO ProjectDetail VALUES(5,3,'ABC');

INSERT INTO ProjectDetail VALUES(6,4,'GHI');

INSERT INTO ProjectDetail VALUES(7,3,'XYZ');

INSERT INTO ProjectDetail VALUES(8,5,'DDC');

Alias: Aliases are often used to make column names more readable

Basic SQL Queries for Practice:

1. Write query to get all employee detail from "EmployeeDetail" table:

ANS: SELECT * FROM EmployeeDetail

2. Write query to get only "FirstName" column from " EmployeeDetail " table

ANS: SELECT FirstName FROM EmployeeDetail

3. Write query to get FirstName in upper case as "First Name".
ANS: SELECT UPPER(FirstName) AS [First Name] FROM EmployeeDetail

4. Write query to get FirstName in lower case as "First Name".
ANS: SELECT LOWER(FirstName) AS [First Name] FROM EmployeeDetail

5. Write query for combine FirstName and LastName and display it as "Name" (also include white space between first name & last name).
ANS:
1st Way - SELECT CONCAT(FirstName+' ',LastName) AS Name FROM EmployeeDetail
2nd Way - SELECT FirstName+' '+LastName AS Name FROM EmployeeDetail

Below are some very good websites to Practice SQL Queries online:

https://sqliteonline.com/
https://www.tutorialspoint.com/execute_sql_online.php
https://rextester.com/l/sql_server_online_compiler

SQL Order By, Where Clause, Like Operator and SQL Wildcards

In this post we will discuss about:

- SQL Order By
- Where Clause
- SQL Like
- SQL Wildcards
- Problems and write SQL queries

Order By

Order By - The ORDER BY keyword is used to sort the result-set in ascending or descending order. The ORDER BY keyword sorts the records in ascending order by default.

Where Clause

The WHERE clause is used to filter records. It is used to extract only those records that fulfill a specified condition. SQL requires single quotes around text values. However, numeric fields should not be enclosed in quotes.
The following operators can be used in the WHERE clause: =, >, <, >=, <=, <>, BETWEEN, LIKE, IN

SQL Wildcards

A wildcard character is used to substitute one or more characters in a string.
Wildcard characters are used with the LIKE operator.

SQL Like

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

% - Represents zero or more characters
Example: Hi% finds Hitendra, Hit, Hi

_ -Represents a single character

Example: V_rma find Verma, varma

[] - Represents any single character within the brackets

Example: h[oa]t finds hot and hat, but not hit

^ - Represents any character not in the brackets

h[^oa]t finds hit, but not hot and hat

- Represents a range of characters

Example: c[a-b]t finds cat and cbt

Queries to Solve

- List the employees in the asc order of their Salaries?

Answer: Select * from EmployeeDetail Order By Salary;

- List the employees in the desc order of their Salaries?

Answer: Select * from EmployeeDetail Order By Salary DESC;

- List the details of the emps in asc order of the FirstName and desc of Department?

Answer: SELECT * from EmployeeDetail Order BY Firstname ASC, Department DESC;

- Select employee detail whose name is "Jane"

Answer: Select * from EmployeeDetail where FirstName='Jane'

- Get all employee detail from Employee table whose "FirstName" start with letter 'r'.

Answer: Select * from EmployeeDetail where FirstName like 'R%'

- Get all employee details from Employee table whose "FirstName" contains 'a'

Answer: Select * from EmployeeDetail where FirstName like '%a%'

- Get all employee details from EmployeeDetail table whose "FirstName" end with 'm'

Answer: Select * from EmployeeDetail where FirstName like '%m'

- Get all employee detail from EmployeeDetail table whose "FirstName" start with any single character between 'a-p'

Answer: Select * from EmployeeDetail where FirstName like '[a-p]%'

Some More SQL Queries on Wildcard, Distinct etc

- Get all employee detail from Employee table whose "FirstName" not start with any single character between 'a-p'

Answer: SELECT * FROM [Employee] WHERE FirstName like '[^a-p]%'

- Get all employee detail from Employee table whose "Gender" end with 'le' and contain 4 letters.

Answer: --The Underscore(_) Wildcard Character represents any single character.

SELECT * FROM [EmployeeDetail] WHERE Gender like '__le' --there are two "_"

- Get all employee detail from Employee table whose "FirstName" start with 'F' and contain 5 letters.

Answer: SELECT * FROM [EmployeeDetail] WHERE FirstName like 'F____' --there are four "_"

- Get all employee detail from Employee table whose "FirstName" containing '%'. ex:-"Fra%nk".

Answer: SELECT * FROM [EmployeeDetail] WHERE FirstName like '%[%]%'
--According to our table it would return 0 rows, because no name containg '%
DISTINCT keyword - The SELECT DISTINCT statement is used to return only distinct (different) values.
The SQL DISTINCT keyword is used in conjunction with the SELECT statement to eliminate all the duplicate records and fetching only unique records.

- Get all unique "Department" from Employee table.

Answer: SELECT DISTINCT(Department) FROM EmployeeDetail

SQL Queries on SQL Like, SQL Wildcards, RTRIM() and LTRIM() functions

- Select all employee detail with First name "Tom" and "Lucy"

Answer: Select * from EmployeeDetail Where FirstName IN ('Tom','Lucy')

- Select all employee detail with First name Not "Frank" and "Jane"

Answer: Select * from EmployeeDetail Where FirstName NOT IN ('Frank','Jane')

- Select first name from "EmployeeDetail" table after removing white spaces from right side

Answer: Select RTRIM(FirstName) AS [First Name] From EmployeeDetail

- Select first name from "EmployeeDetail" table after removing white spaces from left side

Answer: Select LTRIM(FirstName) AS [First Name] From EmployeeDetail
Where EmpId=13

- Select first name from "EmployeeDetail" table prefixed with "Hi

Answer: Select 'Hi '+FirstName from EmployeeDetail

- Get employee details from "EmployeeDetail" table whose Salary greater than 70000

Answer: Select * from EmployeeDetail where Salary>70000

- Get employee details from "EmployeeDetail" table whose Salary less than 70000

Answer: Select * from EmployeeDetail where Salary<70000

- Get employee details from "EmployeeDetail" table whose Salary between 50000 than 60000

Answer: Select * from EmployeeDetail Where Salary Between 40000 AND 60000

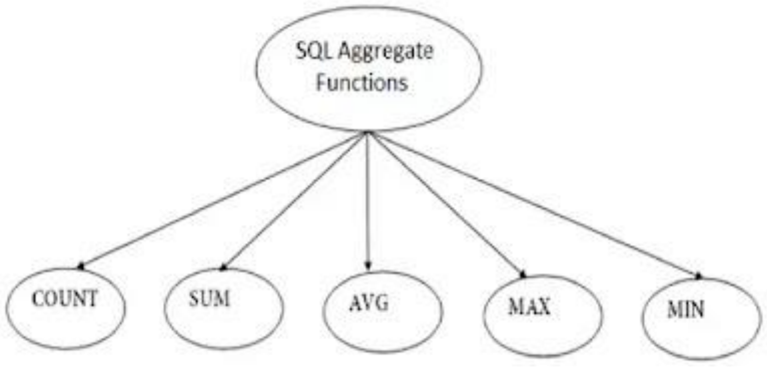
SQL Aggregate Functions and Group By Clause

In this post we will discuss below points:

- SQL Aggregate Functions
- Explanation of each with Example
- Group By Clause
- 10 SQL Queries

Aggregate Function

SQL aggregation function is used to perform the calculations on multiple rows of a single column of a table. It returns a single value.



Count()

COUNT function is used to Count the number of rows in a database table. It can work on both numeric and non-numeric data types.

Count(*): Returns total number of records .
Count(salary): Return number of Non Null values over the column salary.

Sum()

Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.
Example: Select SUM(Salary) from EmployeeDetail.

Avg()

The AVG function is used to calculate the average value of the numeric type. AVG function returns the average of all non-Null values.
Example: Select AVG(Salary) from EmployeeDetail.

Max()

MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.
Example: Select MAX(Salary) from EmployeeDetail.

Max()

MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.
Example: Select MAX(Salary) from EmployeeDetail.

Min()

MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.

Example: Select MIN(Salary) from EmployeeDetail.

Group By Keyword

The GROUP BY clause is a SQL command that is used to group rows that have the same values. The GROUP BY clause is used in the SELECT statement.
The GROUP BY statement is often used with aggregate functions (COUNT() , MAX() , MIN() , SUM() , AVG()) to group the result-set by one or more columns.
Example: SELECT column1, column2
    FROM table_name
    WHERE [conditions]
    GROUP BY column1, column2
    ORDER BY column1, column2

Sample Queries:
Get the highest "Salary" **from** EmployeeDetail **table**.
SELECT MAX(Salary) **As** MaxSalary **FROM** [EmployeeDetail]

Get the lowest "Salary" **from** EmployeeDetail **table**.
SELECT MIN(Salary) **As** MinSalary **FROM** [EmployeeDetail]

Get how many employee exist **in** "EmployeeDetail" **table**.
SELECT COUNT(*) **FROM** [EmployeeDetail]

Write the query **to get** the department **and** department wise total(**sum**) salary **from** "EmployeeDetail" **table**.
SELECT Department, **SUM**(Salary) **AS** [Total Salary] **FROM** [EmployeeDetail] **GROUP BY** Department

Write the query **to get** the department **and** department wise total(**sum**) salary, display it **in** ascending **order** according **to** salary.
SELECT Department, **SUM**(Salary) **AS** [Total Salary] **FROM** [EmployeeDetail] **GROUP BY** Department **ORDER BY SUM**(Salary) **ASC**

Write the query **to get** the department **and** department wise total(**sum**) salary, display it **in** descending **order** according **to** salary.
SELECT Department, **SUM**(Salary) **AS** [Total Salary] **FROM** [EmployeeDetail] **GROUP BY** Department **ORDER BY SUM**(Salary) **DESC**

Write the query **to get** the department, total **no. of** emp **in each** department, total(**sum**) salary **with** respect **to** department **from** "EmployeeDetail" **table**.
SELECT Department, **COUNT**(*) **AS** [Dept Counts], **SUM**(Salary) **AS** [Total Salary] **FROM** [EmployeeDetail] **GROUP BY** Department

Get department wise average salary **from** "EmployeeDetail" **table order by** salary ascending.

SELECT Department, **AVG**(Salary) **AS** [Average Salary] **FROM** [EmployeeDetail] **GROUP BY** Department **ORDER BY AVG**(Salary) **ASC**

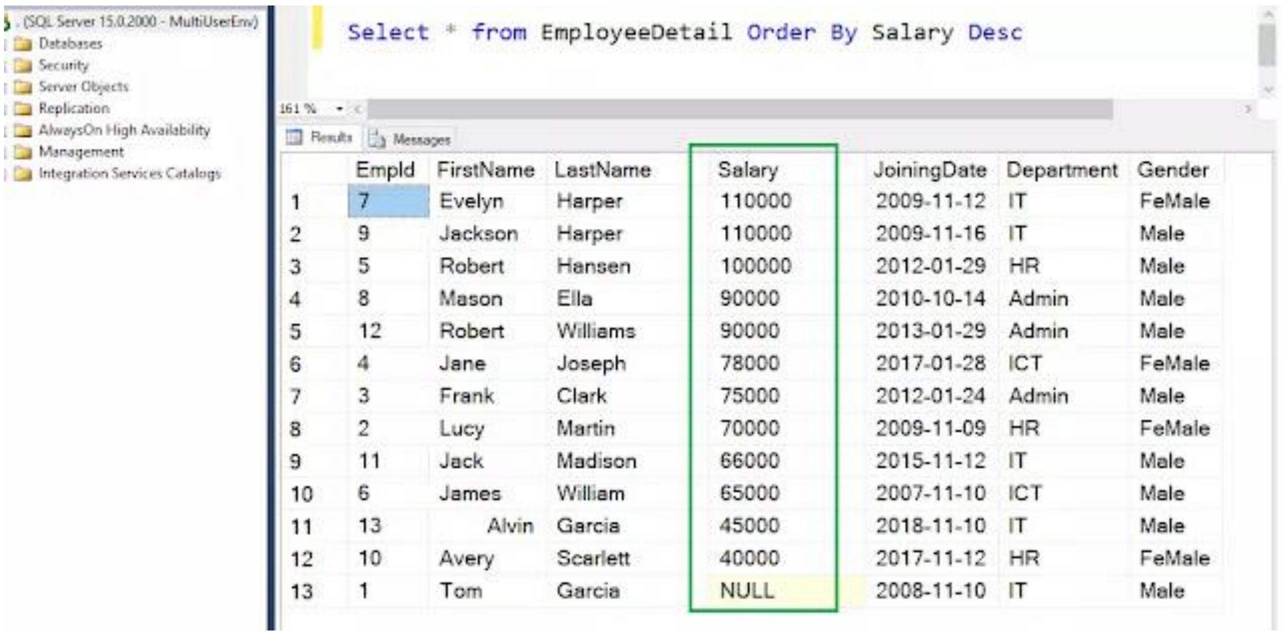
Get department wise maximum salary **from** "EmployeeDetail" **table order by** salary ascending.
SELECT Department, **MAX**(Salary) **AS** [Average Salary] **FROM** [EmployeeDetail] **GROUP BY** Department **ORDER BY MAX**(Salary) **ASC**

Get department wise minimum salary **from** "EmployeeDetail" **table order by** salary ascending.
SELECT Department, **MIN**(Salary) **AS** [Average Salary] **FROM** [EmployeeDetail] **GROUP BY** Department **ORDER BY MIN**(Salary) **ASC**

How to find 2nd Highest Salary | Find Nth Salary | 4 ways to get it

In This tutorial we will discuss about how to get 2nd highest salary in employee table. Below are the different ways:

EmployeeDetail Table:


(SQL Server 15.0.2000 - MultiUserEnv)
Databases
Security
Server Objects
Replication
AlwaysOn High Availability
Management
Integration Services Catalogs
161 %
Results
Messages

```
Select * from EmployeeDetail Order By Salary Desc
```

	EmpId	FirstName	LastName	Salary	JoiningDate	Department	Gender
1	7	Evelyn	Harper	110000	2009-11-12	IT	FeMale
2	9	Jackson	Harper	110000	2009-11-16	IT	Male
3	5	Robert	Hansen	100000	2012-01-29	HR	Male
4	8	Mason	Ella	90000	2010-10-14	Admin	Male
5	12	Robert	Williams	90000	2013-01-29	Admin	Male
6	4	Jane	Joseph	78000	2017-01-28	ICT	FeMale
7	3	Frank	Clark	75000	2012-01-24	Admin	Male
8	2	Lucy	Martin	70000	2009-11-09	HR	FeMale
9	11	Jack	Madison	66000	2015-11-12	IT	Male
10	6	James	William	65000	2007-11-10	ICT	Male
11	13	Alvin	Garcia	45000	2018-11-10	IT	Male
12	10	Avery	Scarlett	40000	2017-11-12	HR	FeMale
13	1	Tom	Garcia	NULL	2008-11-10	IT	Male

Using sub-query (With < Operator):

First, we find the employee with highest salary. To do this we run this query:

Select max(Salary) from EmployeeDetail

This will give us the Maximum Salary; we can further nest this query to a subquery to find the Second Highest Salary. This query will work on MYSQL, ORACLE as well as SQL Server :

Using sub-query (With Not In Operator):
We can also use the NOT IN Clause instead of comparing the salary between two sets.

Using correlated sub-query:

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. The highest salary means no salary is higher than it, the 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.

For each record processed by outer query, inner query will be executed and will return how many records has records has salary less than the current salary. If you are looking for second highest salary then your query will stop as soon as inner query will return 2.

Using TOP Clause keyword:

We can use TOP Command in SQL Server. The sub-query produces a result set containing the salary of employees arranged in decreasing order and fetches TOP 2 records from the set; we use DISTINCT (can be ignored) to segregate duplicate values (if any). Then from RESULT of sub-query, we order the two rows in Ascending order the get the Topmost row which gives us the 2nd Highest Salary.

Please find below all the queries as per above method:

Using sub-query (With < Operator):

2nd Highest Salary
```
Select max(Salary) As Salary from EmployeeDetail where
Salary < (Select max(Salary) from EmployeeDetail
```

3rd Highest Salary
```
Select max(Salary) As Salary from EmployeeDetail where
Salary < (Select max(Salary) from EmployeeDetail Where
Salary < (Select max(Salary) from EmployeeDetail))
```

Using sub-query (With Not In Operator)
```
Select max(Salary) As Salary from EmployeeDetail where
Salary NOT IN (Select max(Salary) from EmployeeDetail)
```

Using correlated sub-query

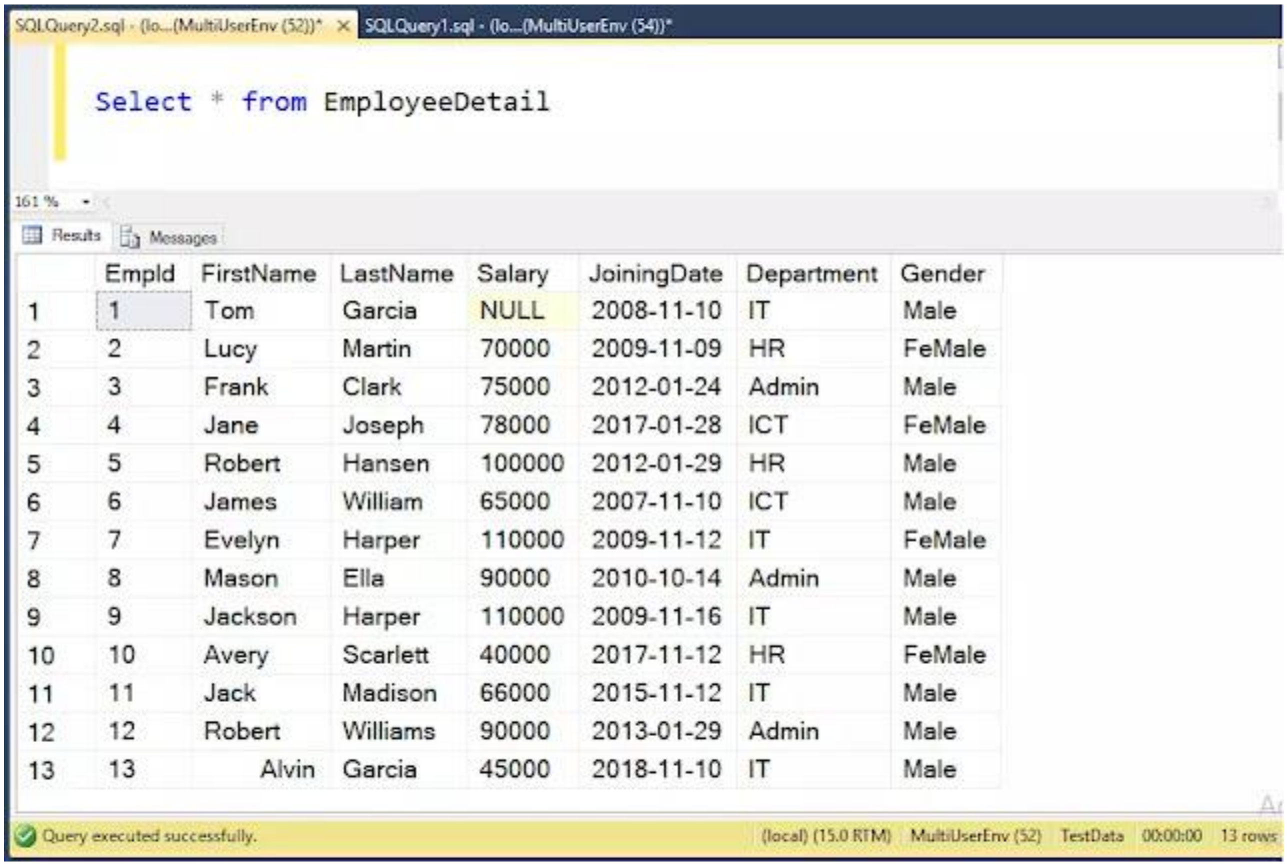
```
Select distinct Salary from EmployeeDetail A
Where 2=(Select count(distinct salary) from EmployeeDetail B where
A.Salary <= b.Salary)
```

Using TOP Clause keyword

```
Select top 1 Salary From
(Select distinct top 2 Salary from
EmployeeDetail Order by Salary Desc)
As Temp Order by Salary
```

How to find 2nd Lowest Salary | Find Nth Lowest Salary | 4 ways to get it

EmployeeDetail table:



```
Select * from EmployeeDetail
```

	EmpId	FirstName	LastName	Salary	JoiningDate	Department	Gender
1	1	Tom	Garcia	NULL	2008-11-10	IT	Male
2	2	Lucy	Martin	70000	2009-11-09	HR	FeMale
3	3	Frank	Clark	75000	2012-01-24	Admin	Male
4	4	Jane	Joseph	78000	2017-01-28	ICT	FeMale
5	5	Robert	Hansen	100000	2012-01-29	HR	Male
6	6	James	William	65000	2007-11-10	ICT	Male
7	7	Evelyn	Harper	110000	2009-11-12	IT	FeMale
8	8	Mason	Ella	90000	2010-10-14	Admin	Male
9	9	Jackson	Harper	110000	2009-11-16	IT	Male
10	10	Avery	Scarlett	40000	2017-11-12	HR	FeMale
11	11	Jack	Madison	66000	2015-11-12	IT	Male
12	12	Robert	Williams	90000	2013-01-29	Admin	Male
13	13	Alvin	Garcia	45000	2018-11-10	IT	Male

Query executed successfully. (local) (15.0 RTM) MultiUserEnv (52) TestData 00:00:00 13 rows

Four ways to get Nth lowest salary in EmployeeDetail table:

```
Select * from EmployeeDetail order by Salary
---------------------1st Method---------------------------
--Using sub-query (With > Operator):
```

```
--2nd Highest Salary
Select min(Salary) As Salary from EmployeeDetail where
Salary > (Select min(Salary) from EmployeeDetail)

--3rd Highest Salary
Select min(Salary) As Salary from EmployeeDetail where
Salary > (Select min(Salary) from EmployeeDetail Where
Salary > (Select min(Salary) from EmployeeDetail))

---------------------2nd Method---------------------------
--Using sub-query (With Not In Operator)
Select min(Salary) As Salary from EmployeeDetail where
Salary NOT IN (Select min(Salary) from EmployeeDetail)

---------------------3rd Method---------------------------
--Using correlated sub-query
Select distinct Salary from EmployeeDetail A
Where 2=(Select count(distinct salary) from EmployeeDetail B
where A.Salary >= b.Salary)

---------------------4th Method----------------------------
--Using TOP Clause keyword
Select top 1 Salary From
(Select distinct top 2 Salary from
EmployeeDetail Order by Salary)
As Temp Order by Salary DESC

/*NOTE: First three method won't consider NULL however 4th method
will consider NULL values*/
```

SQL HAVING Clause

What is having Clause?

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

A HAVING clause in SQL specifies that an SQL SELECT statement must only return rows where aggregate values meet the specified conditions. After the aggregating operation, HAVING is applied, filtering out the rows that don't match the specified conditions.

Why we cannot use where with aggregate function?

We cannot use the WHERE clause with aggregate functions because it works for filtering individual rows. In contrast, HAVING can works with aggregate functions because it is used to filter groups.

Basic Syntax:

SELECT column_names

FROM tableName

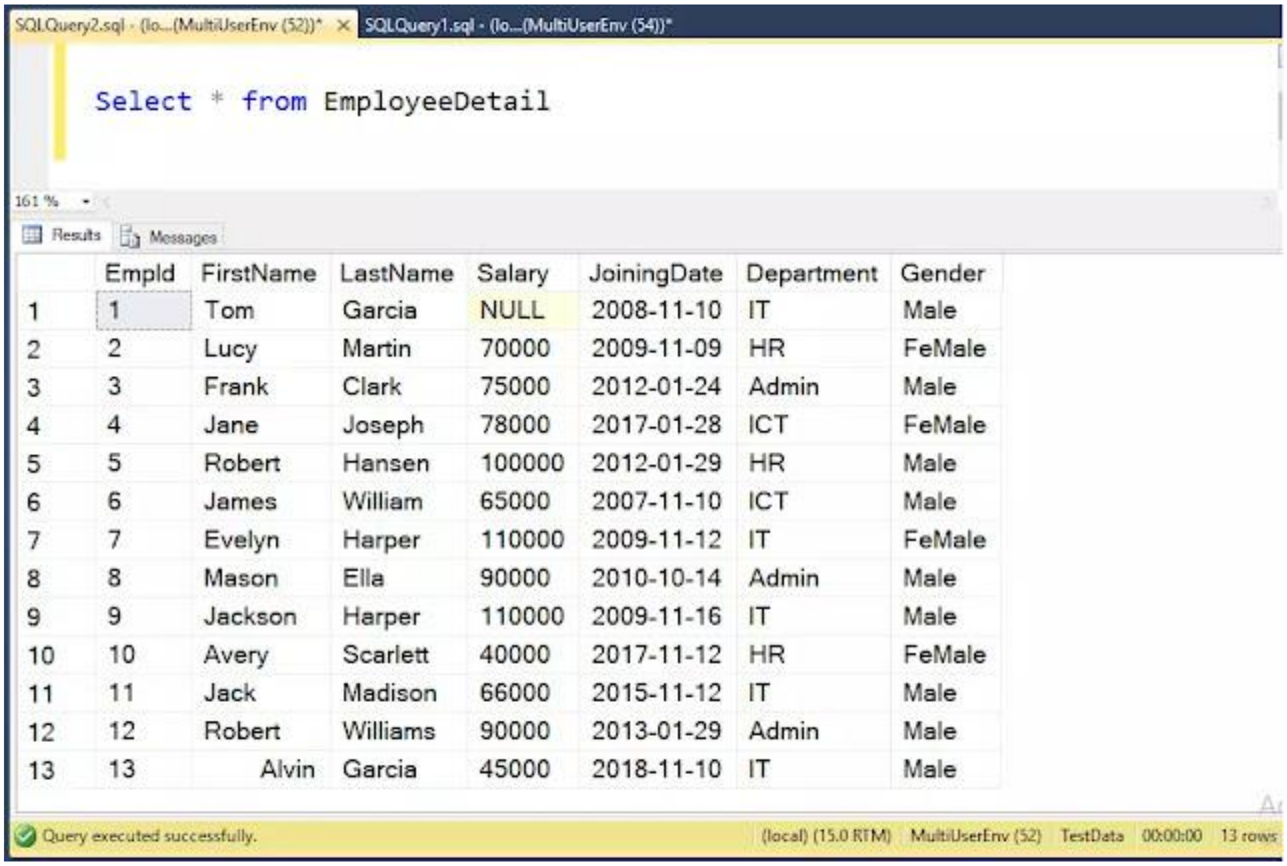
WHERE condition

GROUP BY column_names

HAVING condition

ORDER BY column_names

EmployeeDetail Table:

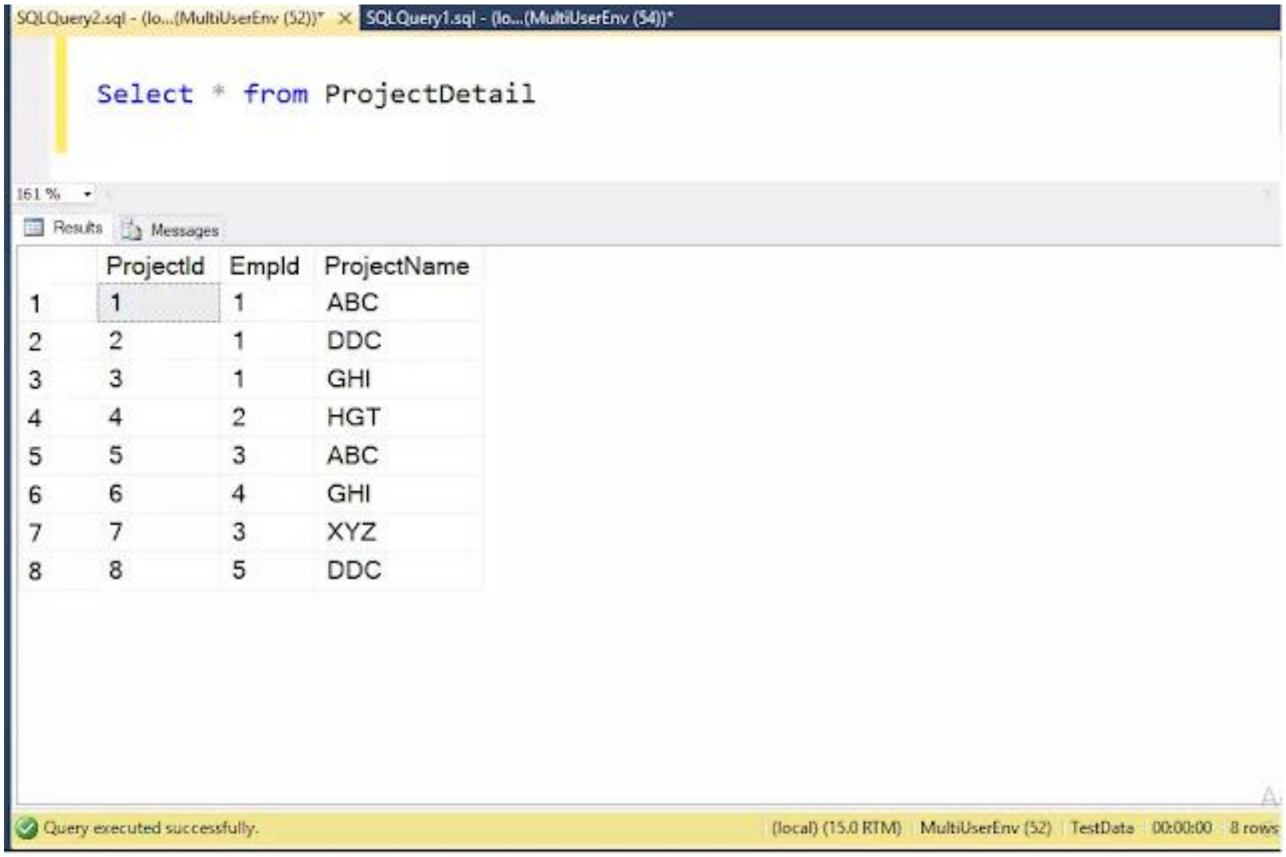


```
Select * from EmployeeDetail
```

	EmpId	FirstName	LastName	Salary	JoiningDate	Department	Gender
1	1	Tom	Garcia	NULL	2008-11-10	IT	Male
2	2	Lucy	Martin	70000	2009-11-09	HR	FeMale
3	3	Frank	Clark	75000	2012-01-24	Admin	Male
4	4	Jane	Joseph	78000	2017-01-28	ICT	FeMale
5	5	Robert	Hansen	100000	2012-01-29	HR	Male
6	6	James	William	65000	2007-11-10	ICT	Male
7	7	Evelyn	Harper	110000	2009-11-12	IT	FeMale
8	8	Mason	Ella	90000	2010-10-14	Admin	Male
9	9	Jackson	Harper	110000	2009-11-16	IT	Male
10	10	Avery	Scarlett	40000	2017-11-12	HR	FeMale
11	11	Jack	Madison	66000	2015-11-12	IT	Male
12	12	Robert	Williams	90000	2013-01-29	Admin	Male
13	13	Alvin	Garcia	45000	2018-11-10	IT	Male

Query executed successfully. (local) (15.0 RTM) MultiUserEnv (52) TestData 00:00:00 13 rows

ProjectDetail Table:


SQLQuery2.sql - (lo...(MultiUserEnv (52))*
SQLQuery1.sql - (lo...(MultiUserEnv (54))*
161 %
Results
Messages

```
Select * from ProjectDetail
```

	ProjectId	EmpId	ProjectName
1	1	1	ABC
2	2	1	DDC
3	3	1	GHI
4	4	2	HGT
5	5	3	ABC
6	6	4	GHI
7	7	3	XYZ
8	8	5	DDC

Query executed successfully. (local) (15.0 RTM) MultiUserEnv (52) TestData 00:00:00 8 rows

Below Sample Queries using Having Clause:

```
/*Write Down the query to fetch the number of employees in each department.
Only include department with more than 2 employees */

SELECT Department, count(EmpID) As NoOfEmploees
FROM EmployeeDetail
GROUP BY Department
HAVING count(EmpID)>2
ORDER BY count(EmpID) DESC

--Write Down the query to fetch Project name assign more than one employee

SELECT ProjectName, count(*) As NoOfEmployees
FROM ProjectDetail
GROUP BY ProjectName
HAVING count(*)>1

--Write Down the query to fetch Employee name(FirstName) assign more than one Project.

SELECT ed.FirstName,Count(ProjectName) NoOfProjects FROM EmployeeDetail ed
INNER JOIN ProjectDetail pd
```

```
ON ed.EmpId=pd.EmpId
GROUP BY FirstName
HAVING count(*)>1

/*Write Down the query to fetch if the Employees 'Frank' or 'Tom' have assigned with
more than 2 Projects*/

SELECT ed.FirstName,Count(ProjectName) NoOfProjects FROM EmployeeDetail ed
INNER JOIN ProjectDetail pd
ON ed.EmpId=pd.EmpId
WHERE ed.FirstName='Frank' OR ed.FirstName='Tom'
GROUP BY FirstName
HAVING count(*)>1
```

SQL Date functions and Formats

SQL Functions:

In this article, we will explore various SQL functions and Convert Date formats to use in writing SQL queries.

SYSDATETIME() - to get current system date

GETUTCDATE() - to get current UTC date

GETDATE() - to get current system date

DATEDIFF() - Diff between two dates

DATEPART() - to get some part of date for example - if you want to retrieve year part (2020) from date 12/09/2020

SQL Convert Date Formats:

We can use the SQL CONVERT() function in SQL Server to format DateTime in various formats.

Syntax for the SQL CONVERT() function is as follows.

SELECT CONVERT (data_type(length)),Date, DateFormatCode)

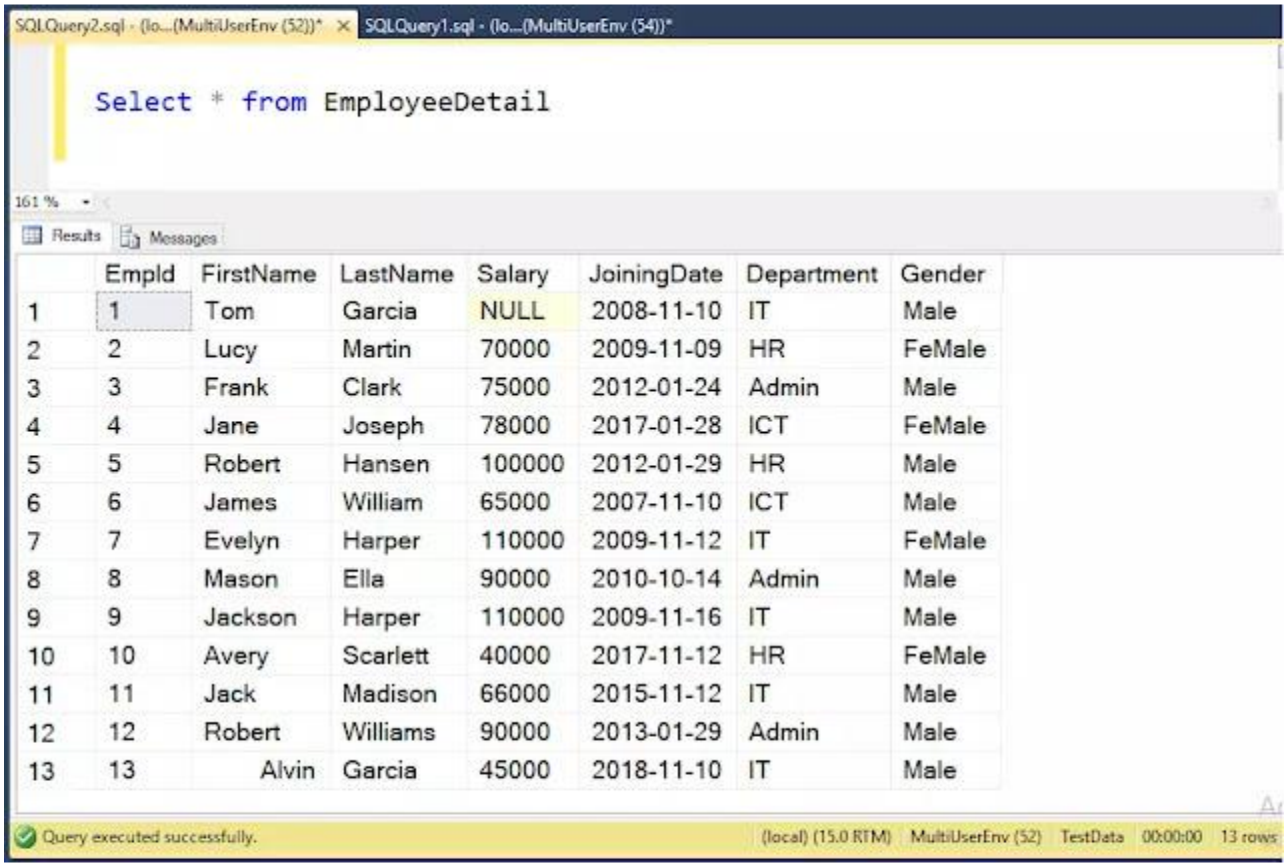
Data_Type: We need to define data type along with length. In the date function, we use Varchar(length) data types

Date: We need to specify the date that we want to convert

DateFormatCode: We need to specify DateFormatCode to convert a date in an appropriate form. We will explore more on this in the upcoming section

Employee Detail table:

Select * from EmployeeDetail



SQLQuery2.sql - (lo...(MultiUserEnv (52))* SQLQuery1.sql - (lo...(MultiUserEnv (54))*

```
Select * from EmployeeDetail
```

	EmpId	FirstName	LastName	Salary	JoiningDate	Department	Gender
1	1	Tom	Garcia	NULL	2008-11-10	IT	Male
2	2	Lucy	Martin	70000	2009-11-09	HR	FeMale
3	3	Frank	Clark	75000	2012-01-24	Admin	Male
4	4	Jane	Joseph	78000	2017-01-28	ICT	FeMale
5	5	Robert	Hansen	100000	2012-01-29	HR	Male
6	6	James	William	65000	2007-11-10	ICT	Male
7	7	Evelyn	Harper	110000	2009-11-12	IT	FeMale
8	8	Mason	Ella	90000	2010-10-14	Admin	Male
9	9	Jackson	Harper	110000	2009-11-16	IT	Male
10	10	Avery	Scarlett	40000	2017-11-12	HR	FeMale
11	11	Jack	Madison	66000	2015-11-12	IT	Male
12	12	Robert	Williams	90000	2013-01-29	Admin	Male
13	13	Alvin	Garcia	45000	2018-11-10	IT	Male

Query executed successfully. (local) (15.0 RTM) MultiUserEnv (52) TestData 00:00:00 13 rows

Exercise Queries:

```
--1. Show "JoiningDate" in "dd mmm yyyy" format, ex- "15 Mar 2020"
SELECT CONVERT(VARCHAR(20),JoiningDate,106) AS ResultDateFormat FROM
EmployeeDetail

--2. Show "JoiningDate" in "yyyy/mm/dd" format, ex- "2013/02/21"
SELECT CONVERT(VARCHAR(20),JoiningDate,111) AS ResultDateFormat FROM
EmployeeDetail

--3. Show only time part of the "JoiningDate". ex- hh:mm: ss
SELECT CONVERT(VARCHAR(20),JoiningDate,108) AS ResultDateFormat FROM
EmployeeDetail

--4. Get only Year part of "JoiningDate".
SELECT DATEPART(YEAR,JoiningDate) AS ResultDateFormat FROM EmployeeDetail
```

```
--5. Get only Month part of "JoiningDate".
SELECT DATEPART(MM,JoiningDate) AS ResultDateFormat FROM EmployeeDetail

--6. Get only Day part of "JoiningDate".
SELECT DATEPART(DD,JoiningDate) AS ResultDateFormat FROM EmployeeDetail

--7. Get system date.
SELECT SYSDATETIME()
SELECT GETDATE()

--8. Get UTC date.
SELECT GETUTCDATE()

/*--9. Get the first name, current date, joiningdate and diff between
current date and joining date in months. */

SELECT FirstName,GETDATE() AS [Current Date], JoiningDate
,DATEDIFF(MM,JoiningDate,GETDATE()) AS TotalMonths FROM EmployeeDetail

/*--10.Get the first name, current date, joiningdate and diff between
current date and joining date in days.*/

SELECT FirstName,GETDATE() AS [Current Date], JoiningDate
,DATEDIFF(DD,JoiningDate,GETDATE()) AS TotalDays FROM EmployeeDetail

--11. Get all employee details from EmployeeDetail table whose joining year
is 2008.
SELECT * FROM EmployeeDetail WHERE DATEPART(YYYY,JoiningDate)='2008'

--12. Get all employee details from EmployeeDetail table whose joining month
is Jan(1).
 SELECT * FROM EmployeeDetail WHERE DATEPART(MM,JoiningDate)='1'

/*--13.Get all employee details from EmployeeDetail table whose joining date
between
"2008-01-01" and "2010-12-01".*/

 SELECT * FROM EmployeeDetail WHERE JoiningDate BETWEEN '2008-01-01'
 AND '2010-12-01'
```

SQL Joins

What is SQL Join?

A SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

SQL JOINS are used to retrieve data from multiple tables. A SQL JOIN is performed whenever two or more tables are listed in a SQL statement.

The SQL Joins clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

Different Types of SQL Joins:

Here are the different types of the JOINs in SQL:

- SQL INNER JOIN (sometimes called simple join):
- SQL LEFT OUTER JOIN (sometimes called LEFT JOIN):
- SQL RIGHT OUTER JOIN (sometimes called RIGHT JOIN):
- SQL FULL OUTER JOIN (sometimes called FULL JOIN):
- SELF JOIN
- CROSS JOIN OR CARTESIAN JOIN

Basic SQL JOIN Syntax

```
SELECT a.table1_column1, a.table1_column2, b.table2_column1
FROM table1 a
JOIN table2 b
ON a.table1_column1=b.table2_column2
WHERE (Condition)
ORDER BY a.table1_column1 DESC
```

What are PRIMARY KEY and FOREIGN KEY?

PRIMARY KEY: Uniquely identified each rows/records in a database table.
FOREIGN KEY: A FOREIGN KEY is used to maintain relationship between two tables. It is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table. The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

Please refer the below video on basics of SQL Joins :

Now lets talk about each and every SQL joins in detail:-
We will take the example of below tables:

SQL INNER JOIN
Returns records that have matching values in both tables.

Returns rows when there is a match in both tables.

SQL LEFT OUTER JOIN
Returns all rows from the left table, even if there are no matches in the right table.
This type of join returns all rows from the LEFT hand table specified in the ON condition and only those rows from the other table where the joined fields are equal.
Returns all records from the left table, and the matched records from the right table.
SQL RIGHT OUTER JOIN
Returns all rows from the right table, even if there are no matches in the left table.
Returns all records from the right table, and the matched records from the left table.
SQL FULL OUTER JOIN
FULL JOIN creates the result-set by combining result of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both the tables. The rows for which there is no matching, the result-set will contain NULL values.
SQL SELF JOIN
The self join, as its name implies, joins a table to itself.
Self Join is used to join a table to itself as if the table were two tables.

SQL SELF JOIN Syntax
```
SELECT a.column_name, b.column_name...
FROM table1 a, table1 b
WHERE a.common_field = b.common_field

SELECT a.column_name, b.column_name...
FROM table1 a
JOIN table1 b
ON a.common_field = b.common_field
```
SQL CARTESIAN JOIN OR CROSS JOIN
Returns the Cartesian product of the sets of records from the two or more joined tables.
The CROSS JOIN is used to generate a paired combination of each row of the first table with each row of the second table. This join type is also known as Cartesian join.

Real World Example:
Suppose that we are sitting in a restaurant and we decide to order Lunch. Shortly, we will look at the menu and we will start thinking of which meal and drink combination could be more tastier. Our brain will receive this signal and begin to generate all meal and drink combinations.

Syntax:
```
SELECT ColumnName_1, ColumnName_2, ColumnName_N
FROM [Table_1]
CROSS JOIN [Table_2]
```