**TDD Advantages:**

* **Early bug notification.**
* Developers test their code but in the database world, this often consists of manual tests or one-off scripts. Using TDD you build up, over time, a suite of automated tests that you and any other developer can rerun at will.
* **Better Designed, cleaner and more extensible code.**
* It helps to understand how the code will be used and how it interacts with other modules.
* It results in better design decision and more maintainable code.
* TDD allows writing smaller code having single responsibility rather than monolithic procedures with multiple responsibilities. This makes the code simpler to understand.
* TDD also forces to write only production code to pass tests based on user requirements.
* **Confidence to Refactor**
* If you refactor code, there can be possibilities of breaks in the code. So having a set of automated tests you can fix those breaks before release. Proper warning will be given if breaks found when automated tests are used.
* Using TDD, should results in faster, more extensible code with fewer bugs that can be updated with minimal risks.
* **Good for teamwork**
* In the absence of any team member, other team members can easily pick up and work on the code. It also aids knowledge sharing, thereby making the team more effective overall.
* **Good for Developers**
* Though developers have to spend more time in writing TDD test cases, it takes a lot less time for debugging and developing new features. You will write cleaner, less complicated code.

**Drawbacks of TDD (Test Driven Deveolpment):**

**1. Slow process**

If you begin TDD, you'll get the sensation that you simply need extended time for straight forward implementations. You would like to believe the interfaces, write the test code, and run the tests bef ore you finally start writing the code.

**2. All the members of a team got to do it**

As TDD influences the planning of code, it's recommended that all team members use TDD or nobody the least. Additionally, to the present, it's sometimes difficult to justify TDD to the management because they often have the sensation that implementing the latest features takes longer if developers write code that will not find them within the product half the time. It helps if the entire team agrees on the importance of unit tests.

**3. Tests got to be maintained when requirements change**

Probably, the strongest argument against TDD is that the tests need to be maintained because the code has got to. Whenever requirements change, you would like to vary the code and tests. But you're working with TDD. This suggests that you simply got to change the tests first and then make the tests pass. So, this disadvantage is the same as before when writing code that takes an extended time takes a long time.