eBook

MSRcosmos
PROMISE DELIVERED

DATA ENGINEERING
A GUIDE FOR TOP BEST PRACTICES

TABLE OF CONTENTS

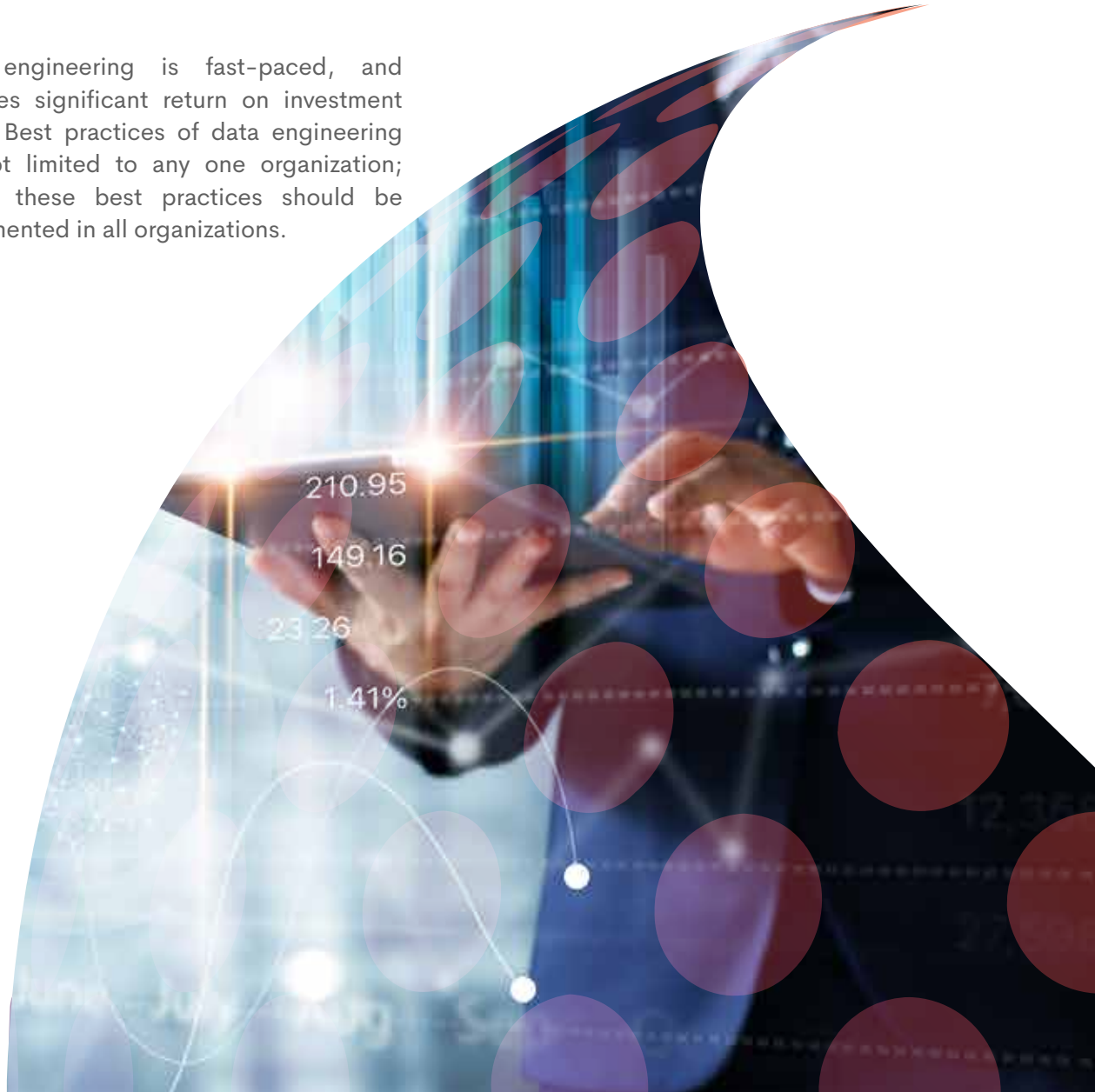
INTRODUCTION

Data engineering is one of the key steps that can help your business become more competitive in the data-driven economy. It's not just about collecting and storing data; it's also about transforming it, cleaning it, extracting additional information from it, and making sense of it all.

The data engineering process, if done right and with the right people, can be one of the most profitable and powerful tools in an organization's arsenal. Data is the new digital dominator, and as a result, organizations that embrace and leverage it effectively will almost certainly find an advantage over their competitors.

Data Engineering process involves putting raw data into a structured format, and sometimes wringing as much value out of it as possible to reduce the amount of data or make it useful for machine learning models, artificial intelligence systems like chatbots, or even applications. The result is a deployed solution that doesn't require external resources like support staff, but enables human users to access their information more easily and quickly.

Data engineering is fast-paced, and provides significant return on investment (ROI). Best practices of data engineering are not limited to any one organization; rather, these best practices should be implemented in all organizations.



13 BEST PRACTICES FOR DATA ENGINEERING

1. TEST YOUR DATA

Test your data before you begin the story, and make sure that you understand the correct format of the data. You can use a tool such as Excel or Python to test your data. If possible, make sure that the data is provided in a structured way and that it contains enough information to be able to do the analysis.

2. TEST YOUR ETL SCRIPTS

ETL (Extract, Transform, Load) is a term used to describe the process of extracting data from one source and transforming it into another format. ETL can be done manually or with the help of tools like MapReduce, Pig or Hadoop. These tools make it easier to create custom ETL processes by combining multiple steps into one command line.

The code should be tested before deployment as there are many possible errors that can occur during runtime, such as exceptions thrown from the code or invalid input parameters. These errors can lead to outages or data loss if not caught in time. When you do find errors, make a step-by-step plan on properly handling them and failures in your ETL scripts.

3. PRACTICE DATA VERSIONING

This is an essential practice for data engineering, and it is important to understand the difference between a system and a dataset. A system is something that is made up of multiple pieces of software, hardware and data sources. A dataset is a group of related systems that are used together to solve one or more problems.

Data versioning means that you create versions of your datasets so that you can easily create new versions or use older versions in new ways. You should never delete old versions unless they are no longer needed or relevant to your current use cases.

You can also use versioning to audit your code base and make sure that you don't have any anomalies in your data.

4. DON'T USE SINGLE POINTS OF FAILURE IN YOUR ARCHITECTURE

When building an application or service, it is important to have multiple layers of redundancy in place.

The more layers of redundancy you have, the less likely it is that something will go wrong with your system and cause it to fail or become unavailable for customers. Having multiple layers of redundancy also helps with performance testing and troubleshooting issues if something does happen

5. DEFINE CLEAR SUCCESS AND FAILURE CRITERIA

A clear definition of success metrics will help you define what is expected from each stage of your data engineering process.

For example, it is important to define the outcome of an ETL process as well as the status of data cleansing activities like cleansing bad data or cleaning duplicates.

6. USE A FAULT-TOLERANT MESSAGING SYSTEM

A messaging system helps you communicate both internally and externally with other departments in your organization about what is happening with your project.

It also helps you track progress and identify problems early on so that they can be resolved quickly and efficiently. The messaging system too is prone to failures that you need to be prepared for.

7. AVOID HARD-CODING VARIABLES AND PATHS.

There have been instances where engineers use hard-coding variables and paths without commenting an explanation or reference. This makes the code difficult to work with in the future.

For example, you see a script written to extract data from an RDBMS table but only belonging to some people from a department of a company. Maybe those people were to be shifted to some other department or maybe they were the best performing employees or any other reason.

When those specific employee ids have been coded, the specific reason must be mentioned. If not, it is best to avoid hard-coding.

8. WRITE CODE USING MODULAR DESIGN

It is always better to keep things in separate modules even if merging business logic and technical functions seem handy initially. For the long run, having separate modules will be very beneficial.

You might want to make a different package for the common functionalities in order to reuse them, in addition to having separate packages for different groups of functions. When it comes to reusability, the common as well as the distinct packages serve well.

9. MAKE SEARCHABILITY A PRIORITY

While designing architecture for your system, ensure that the desired data is easily searchable and retrievable.

You could set expectations about data request workflow, define the workflow according to timelines, create a template for requests, and even automate requests that are repetitive in nature. This will help you manage your data requests well and keep searchability a priority.

When these initial steps are taken care of, you may measure and improve the workflow as and when required on the go.

10. BUILD AND MAINTAIN NOTEBOOKS

Keeping a record of all the applied processes and their results will help in sharing them quickly and conveniently. The records will come in handy when you are looking to apply the same processes or similar ones in a different system. It will help you generate an idea of what results could be expected.

Sharing these documentations will, needless to say, be of great help to those working on the same systems after you. Even you could use them as your future references.

11. WRITE TESTS FOR ALL THE CODES

Testing all the codes whether simple or complex is essential to avoid confusions and complications in the future. Write tests for all the functions, use a sample data set to run it through the system. Once you get the output, compare it with your expected output. If the output matches your expectation, record the results; if not, make amends and run the tests again. Do not avoid testing the basic functions, no matter how simple.

12. OPTIMIZE AND TEST

Make it habitual to test for your systems' performance early and often. Optimize for performance as you go. Step-wise testing and optimization will help in boosting the performance of your system.

By using indexing, avoiding loops, defragmenting data, optimizing queries and memory and using several other tips, you can make your system better.

13. STRUCTURE FOR ANALYSIS

The goal of any business is to stay ahead in competition and keep their return on investment increasing, or at least steady. Analytics and reporting play a crucial role in the achievement of the same. If systems are structured and made to evolve keeping analytics in mind, it will prove to be beneficial for the organization.

Building systems that collect, validate and prepare high-quality data for analysis and reporting drive better decision making.

CONCLUSION

Succinctly, best data engineering practices comprise of keeping the codes simple, testing all the functions, being prepared for failure, documenting all designs for future reference, and structuring systems keeping the end goal in mind. Always keep in mind, to test more than expected inputs and outputs and to be consistent with metrics and logging.

With all the above discussed practices, one can be sure of making progress in business and putting raw data to good use. These will help you create a data engineering pipeline that is both robust and secure.