Architecture Design

Analyzing Amazon Sales Data

Written By	Amit Patil
Document Version	0.1
Last Revised Date	

DOCUMENT CONTROL

Change Record:

VERSION	DATE	AUTHOR	COMMENTS
0.1	06-March-2023	Amit Patil	Introduction and architecture defined
			Architecture & Architecture description appended and updated.

Reviews:

Approval Status:

VERSION	REVIEW DATE	REVIEWED BY		APPROVED BY	COMMENTS

Contents

1. Introduction

1.1 What is Architecture design document?

Any software needs the architectural design to represents the design of software. IEEE defines architectural design as “the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.” The software that is built for computer-based systems can exhibit one of these many architectures.

Each style will describe a system category that consists of :

- A set of components (eg: a database, computational modules) that will perform a function required by the system.
- The set of connectors will help in coordination, communication, and cooperation between the components.
- Conditions that how components can be integrated to form the system.
- Semantic models that help the designer to understand the overall properties of the system.

1.2 Scope

Architecture Design Document (ADD) is an architecture design process that follows a step-by-step refinement process. The process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the design principles may be defined during requirement analysis and then refined during architectural design work.

2. Architecture

2.1 Matplotlib Architecture

Matplotlib is a Python 2D plotting library that allows users to create a wide range of static, animated, and interactive visualizations in Python. It provides a high-level interface for drawing attractive and informative statistical graphics.

2.2 Components of Matplotlib

Matplotlib has several components that work together to create a visualization. These components are as follows:

2.2.1 Figure

A Figure is the top-level container for all the plot elements in Matplotlib. It can contain multiple Axes objects, which in turn contain the plot elements.

2.2.2 Axes

Axes are the individual plot objects that contain the data visualization elements such as lines, bars, and points. A Figure can contain multiple Axes objects, and each Axes can have a different set of data visualization elements.

2.2.3 Axis

Axes are the number-line objects that provide ticks, tick labels, and axis labels. Each Axis is associated with an Axes object and can have a different scale.

2.2.4 Artist

Artists are the individual plot elements such as lines, bars, and points that are drawn on the Axes object.

2.2.5 Back-end

The Back-end is the interface between Matplotlib and the operating system. It handles the low-level details of drawing the plot on the screen or saving it to a file.

2.3 Matplotlib Communication Flow

The communication flow in Matplotlib is as follows: the user creates a Figure object, which contains one or more Axes objects. The user then adds Artist objects to the Axes object, which draws the visualization. Finally, the user saves the visualization to a file or displays it on the screen.

Deployment

3.1 Installation

Matplotlib can be installed using pip, the Python package manager, or through Anaconda, the Python distribution. Once installed, the user can import Matplotlib in a Python script using the following command:

python

Copy code

import matplotlib.pyplot as plt

3.2 Configuration

Matplotlib can be configured using the rcParams dictionary. This dictionary contains key-value pairs that control the default settings of the plot elements such as font size, line width, and color. The user can modify the rcParams dictionary to customize the appearance of the visualization.

To create a deployment architecture for this project, we will consider the usage and scalability requirements. Since the data size and complexity are moderate, we can start with a simple architecture that can be scaled up if needed.

3.3 Usage

The project is intended for internal use by the sales team of the company. The charts will be accessed through a local computer and will not be publicly available. The expected number of concurrent users is low to moderate.

3.4 Architecture

For this project, we will use a single-node architecture with the following components:

Local computer: This is the computer where the charts will be accessed from.

Python environment: The project will be developed in Python using the Matplotlib library for data visualization.

Data storage: The data will be stored in a CSV file.

Matplotlib charts: The charts will be generated using the Matplotlib library and will be displayed on the local computer.

3.5 Deployment

The deployment process will involve the following steps:

- Install Python and Matplotlib on the local computer.
- Download the project files and save them to a local directory.
- Install any required Python libraries (e.g., Pandas, Seaborn) using pip.
- Open the project file in a Python IDE (e.g., PyCharm) and run the code.
- The charts will be generated and displayed on the local computer.

3.6 Scalability

If the usage requirements increase, we can consider scaling up the architecture by using a more powerful computer or moving the data to a cloud-based storage solution. We can also explore using more advanced data visualization libraries (e.g., Plotly, Bokeh) for more interactive and dynamic charts.