High Level Design (HLD)

Analyzing Swiggy: Bangalore delivery outlet data

By
Mamta M. Natu.

2 High Level Documentation

Document Version Control

Date issued	Version	description	Author
25 February 2023	1.0	First Version of Complete HLD	Mamta Natu

3 High Level Documentation

Contents:

Abstract

The online food ordering market includes foods prepared by restaurants, prepared by independent people, and groceries being ordered online and then picked up or delivered. The first online food ordering service, World Wide Waiter (now known as Waiter.com), was founded in 1995. Online food ordering is the process of ordering food from a website or other application. The product can be either ready-to-eat food or food that has not been specially prepared for direction consumption.

Bangalore, the IT hub of the country, is the city that is a home to millions of young techies staying far away from their homes. If you are also one of those, who does not have a proper functional kitchen, you can understand the pain of missing out on your favourite food. The services of food delivery in Bangalore make it easy for you to taste the delicious dishes that this city has to offer. You will not have a dearth of places to order food online, as there are many restaurants offering online food delivery in Bangalore.

Analyzing Bangalore city swiggy data for Finding key metrics and factors and show the meaningful relationships between attributes.

4 High Level Documentation

1 Introduction

1.1 Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual
for how the modules interact at a high level.

The HLD will:

- Present all of the design aspects and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include design features and the architecture of the project
- List and describe the non-functional attributes like:

>security
>Reliability
>Maintainability
>Portability
>Reusability
>Application compatibility
>Resource utilization
>Serviceability

1.2 Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

5 High Level Documentation

2. General Description

2.1 Product Perspective & Problem Statement

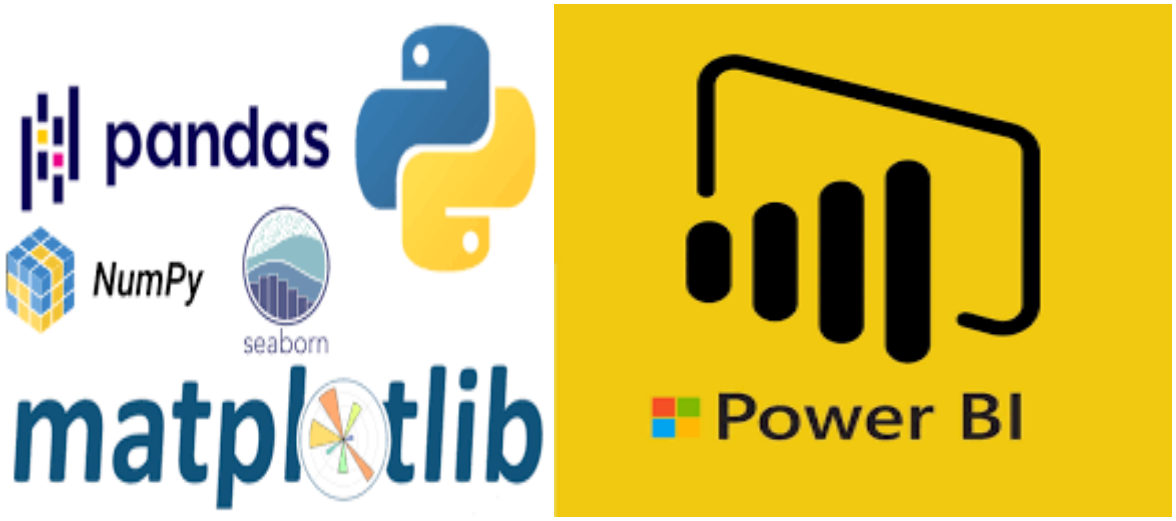
Ordering food online has an important reflection of the economy, and ordering food online is of
great interest to customers, sellers and to swiggy. In this project, we analysed the given swiggy outlets data of Bangalore city.

The objective of the project is: -

1) Do ETL: Extract-Transform-Load the dataset and find some information from this large data. This is form of data mining.
2) To find key metrics and factors and to show the meaningful relationships between attributes
3) To perform data visualization techniques to understand the insight of the data.
4) Project aims to apply Python, R, Tableau, Power BI, Alteryx or you can use any tools and techniques as per convenience and valid imagination in finding solutions.
or Power BI to get a visual understanding of the data.

2.2 Tools used

> **Excel**: To get data from excel sheet
> **python** libraries such as NumPy, Pandas, matplotlib, seaborn used for doing Exploratory Data.
> Analysis **PowerBI** are used to build the whole framework.

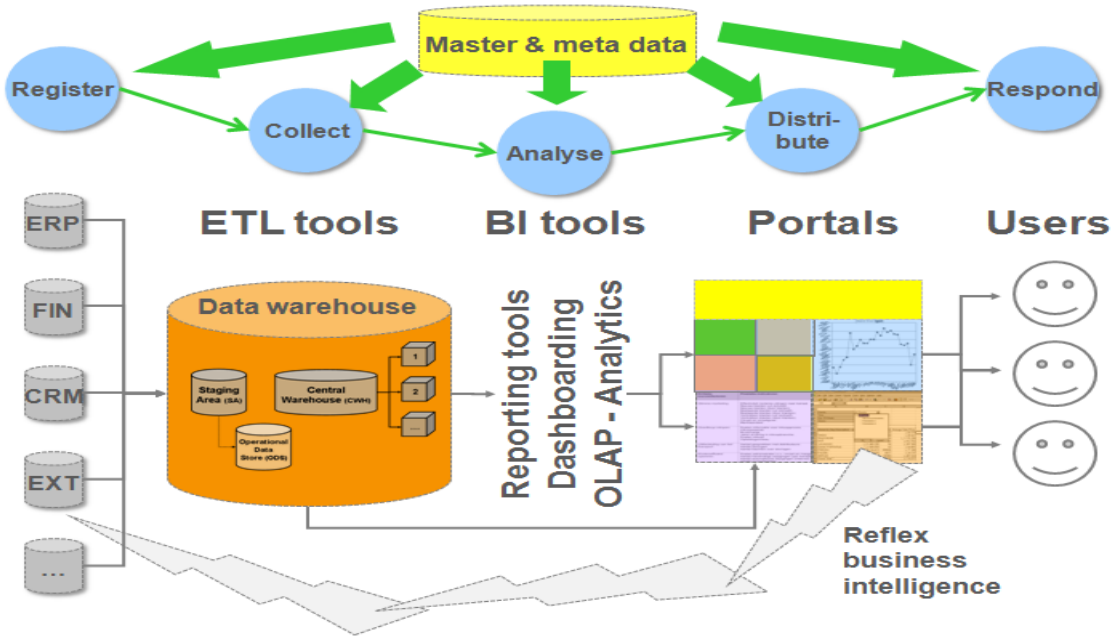


6 High Level Documentation

3 Design Details

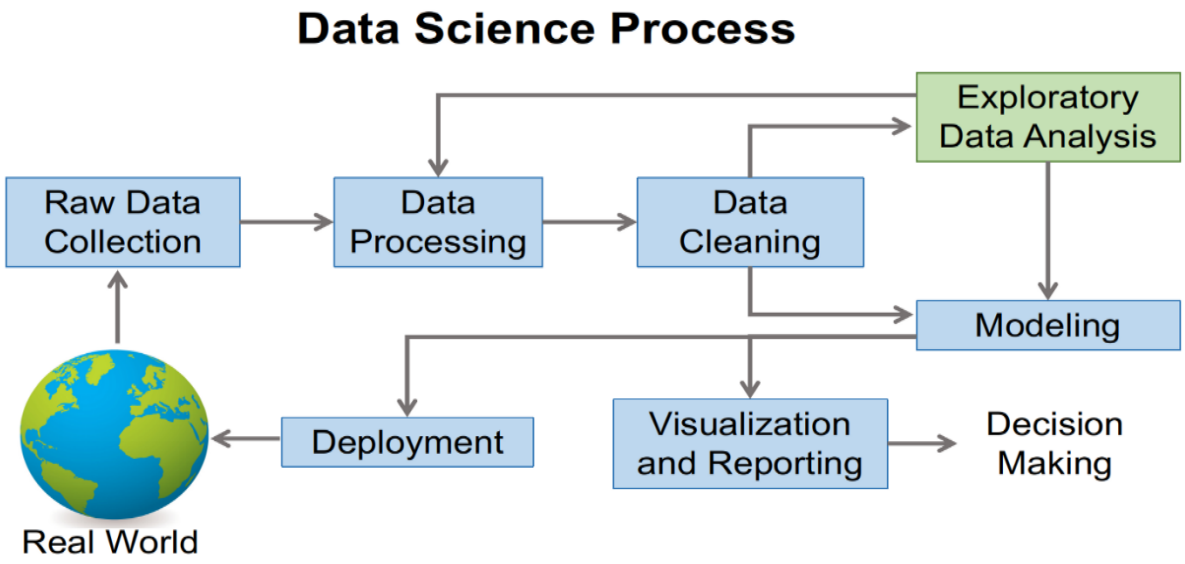
3.1(a) Functional Architecture

Figure 1: Functional Architecture of Business Intelligence



3.1(b) Functional Architecture

Figure 2: Functional architecture of python Exploratory Data analysis and decision making in data science projects.



Exploratory Data Analysis (EDA) Steps with Python

To do Exploratory Data Analysis in Python, we need some python libraries such as NumPy, Pandas, and Seaborn. The last two libraries will be used for visualisation.

1. Check data shape (number of Rows & Columns)
2. Check each data type of columns and missing values
3. Splitting values
4. Change the data type
5. Check the percentages of missing value
6. Summary Statistics
7. Check value counts for a specific column
8. Check duplicate values and deal with it
9. See the data distribution and data anomaly
10. Check the correlation between variables in the data

3.2 Optimization

Optimizing the data model

The data model supports the entire visualization experience. Data models are either external-hosted or internal-hosted, and in Power BI they are referred to as *datasets*. It's important to understand your options, and to choose the appropriate dataset type for your solution. There are three dataset modes: Import, Direct Query, and Composite. For more information, see Datasets in the Power BI service, and Dataset modes in the Power BI service.

Optimizing visualizations

Power BI visualizations can be dashboards, Power BI reports, or Power BI paginated reports. Each has different architectures, and so each has their own guidance.

Dashboards

It's important to understand that Power BI maintains a cache for your dashboard tiles—except live report tiles, and streaming tiles. If your dataset enforces dynamic row-level security (RLS), be sure to understand performance implications as tiles will cache on a per-user basis.

When you pin live report tiles to a dashboard, they're not served from the query cache. Instead, they behave like reports, and make queries to v-cores on the fly.

As the name suggests, retrieving the data from the cache provides better and more consistent performance than relying on the data source. One way to take advantage of this functionality is to have dashboards be the first landing page for your users. Pin often-used and highly requested visuals to the dashboards. In this way, dashboards become a valuable "first line of defence", which delivers consistent performance with less load on the capacity. Users can still click through to a report to analyze details.

For Direct Query and live connection datasets, the cache is updated on a periodic basis by querying the data source. By default, it happens every hour, though you can configure a different frequency in the dataset settings. Each cache update will send queries to the underlying data source to update the cache. The number of queries that generate depends on the number of visuals pinned to dashboards that rely on the data source. Notice that if row-level security is enabled, queries are generated for each different security context.

For example, consider there are two different roles that categorize your users, and they have two different views of the data. During query cache refresh, Power BI generates two sets of queries.

Power BI reports

There are several recommendations for optimizing Power BI report designs. Note
When reports are based on a Direct Query dataset, for additional report design optimizations, apply the most restrictive filters.
The more data that a visual need to display, the slower that visual is to load. While this principle seems obvious, it's easy to forget. For example: suppose you have a large dataset. Atop of that dataset, you build a report with a table. End users use slicers on the page to get to the rows they want—typically, they're only interested in a few dozen rows.
A common mistake is to leave the default view of the table unfiltered—that is, all 100M+ rows. The data for these rows loads into memory and is uncompressed at every refresh. This processing creates huge demands for memory. The solution: use the "Top N" filter to reduce the max number of items that the table displays. You can set the max item to larger than what users would need, for example, 10,000. The result is the end-user experience doesn't change, but memory use drops greatly. And most importantly, performance improves.
A similar design approach to the above is suggested for every visual in your report. Ask yourself, is all the data in this visual needed? Are there ways to filter the amount of data shown in the visual with minimal impact to the end-user experience? Remember, tables in particular can be expensive.

Limit visuals on report pages

The above principle applies equally to the number of visuals added to a report page. It's highly recommended you limit the number of visuals on a particular report page to only what is necessary. Drill through pages and report page tooltips are great ways to provide additional details without jamming more visuals onto the page.

Evaluate custom visual performance

Be sure to put each custom visual through its paces to ensure high performance. Poorly optimized Power BI visuals can negatively affect the performance of the entire report.

Power BI paginated reports

Power BI paginated report designs can be optimized by applying best practice design to the report's data retrieval. For more information, see Data retrieval guidance for paginated reports.
Also, ensure your capacity has sufficient memory allocated to the paginated reports workload.

9 High Level Documentation

4 KPIs:

4.1 KPIs (Key Performance Indicators)

Key indicators displaying a summary of the swiggy data and its relationship with different metrics

1. Relation between location and number of cuisine types and number of cuisines.
2. Relation between cuisines and number of ratings.
3. influence of location on ratings and cost for two persons.
4. Relationship between cost for two persons and ratings.
5. Top 5 cuisine type and their ratings and cost for two.
6. Shops having maximum number of cuisines types at different locations in city.

5 Deployment

Deployment pipelines is designed as a pipeline with three stages:

1)**Development**

This stage is used to design, build, and upload new content with fellow creators. This is the first stage in deployment pipelines.

2)**Test**

You're ready to enter the test stage after you've made all the needed changes to your content. You upload the modified content so it can be moved to this test stage. Here are three examples of what can be done in the test environment:

Share content with testers and reviewers

Load and run tests with larger volumes of data

Test your app to see how it will look for your end users

3)**Production**

After testing the content, use the production stage to share the final version of your content with business users across the organization.