High Level Design

Analysing Swiggy: Bangalore delivery outlet Ankita chetan chaudhari

[AUTHOR NAME]

[AUTHOR N

Contents

[Abstract 4](#_Toc9802)

[1. Introduction 5](#_Toc9803)

[1.1 Why this High-Level Design Document? 5](#_Toc9804)

[1.2 Scope 6](#_Toc9805)

[2. General Description 7](#_Toc9806)

[2.1 Product Perspective & Problem Statement 7](#_Toc9807)

[2.2 Tools used 7](#_Toc9808)

[3. Design Details 8](#_Toc9809)

[3.1 Functional Architecture 8](#_Toc9810)

[3.2 Optimization 9](#_Toc9811)

[4. KPIs & Charts 10](#_Toc9812)

[4.1 KPIs (Key Performance Indicators) 10](#_Toc9813)

[4.2 Charts 10](#_Toc9814)

[5. Deployment 11](#_Toc9815)

# Abstract

The Swiggy analysis Food Sales report describe the sales of the various range of products in Domestic as well as in International countries. This report may help to take necessary steps after seeing the key insights and analysis. The Detailed analysis of the Sales of Swiggy by using the Power BI tool which is capable of showcasing key insights of the sales from the Given data.

[AUTHOR NAME

[AUTHOR NAME

# Introduction

## 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:

o Security o Reliability o Maintainability o Portability o Reusability o Application compatibility o Resource utilization o Serviceability

## 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.

[AUTHOR

# General Description

## Product Perspective & Problem Statement

Sales management has gained importance to meet increasing competition and the need for improved methods of distribution to reduce cost and to increase profits. Sales management today is the most important function in a commercial and business enterprise.

Do ETL: Extract-Transform-Load some Swiggy dataset and find for me Sales-trend -> location wise, restaurant wise, food items wise.

## Tools used

Business Intelligence tool Power BI is used to build the whole framework.



[AUTHOR NAME]

# Design Details

## Functional Architecture



Problem



Evaluation



Data Gathering



Data



Pre-processing



Data Analysis

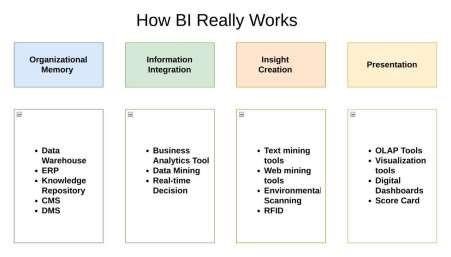


Data



Visualization

Figure 1: Functional Architecture of Business Intelligence



[AUTHOR NAME

## Optimization

#### our data strategy drives performance

* Minimize the number of fields
* Minimize the number of records
* Optimize extracts to speed up future queries by materializing calculations, removing columns and the use of accelerated views

#### Reduce the marks (data points) in our view

* Practice guided analytics. There’s no need to fit everything you plan to show in a single view. Compile related views and connect them with action filters to travel from overview to highly granular views at the speed of thought.
* Remove unneeded dimensions from the detail shelf.
* Explore. Try displaying your data in different types of views.

#### Limit our filters by number and type

* Reduce the number of filters in use. Excessive filters on a view will create a more complex query, which takes longer to return results. Double-check your filters and remove any that aren’t necessary.
* Use an include filter. Exclude filters load the entire domain of a dimension, while include filters do not. An include filter runs much faster than an exclude filter, especially for dimensions with many members.
* Use a continuous date filter. Continuous date filters (relative and range-of-date filters) can take advantage of the indexing properties in your database and are faster than discrete date filters.
* Use Boolean or numeric filters. Computers process integers and Booleans (t/f) much faster than strings.
* Use parameters and action filters. These reduce the query load (and work across data sources).

#### Optimize and materialize your calculations

* Perform calculations in the database • Reduce the number of nested calculations. • Reduce the granularity of LOD or table calculations in the view. The more granular the calculation, the longer it takes. o LODs - Look at the number of unique dimension members in the calculation. o Table Calculations - the more marks in the view, the longer it will take to calculate.
* Where possible, use MIN or MAX instead of AVG. AVG requires more processing than MIN or MAX. Often rows will be duplicated and display the same result with MIN, MAX, or AVG. • Make groups with calculations. Like include filters, calculated groups load only named members of the domain, whereas Power bi group function loads the entire domain.
* Use Booleans or numeric calculations instead of string calculations. Computers can process integers and Booleans (t/f) much faster than strings. Boolean>Int>Float>Date>Date Time>String

## 4.Charts

Charts displaying an understanding of Swiggy Food Sales Data

1. Count of Shop name by location

2. Count of Cuisine by Location

3. Count of Location by Cuisine

4. Shop name by rating (Matrix Table)

5. Cost of two

[AUTHOR NAME]

### [AUTHOR NAME]