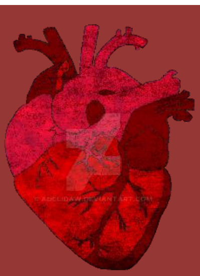


High Level Design (HLD)

Heart Disease Diagnostic Analysis

Revision Number – 1.3

Last Date of Revision – 17/12/2021

William Johnson

Document Version Control

Date Issued	Version	Description	Author
20/11/2021	1.0	Abstract, Introduction,	William Johnson
01/12/2021	1.1	General Description	William Johnson
11/12/2021	1.2	Design Detail	William Johnson
17/12/2021	1.3	Final Revision	William Johnson

Contents

Abstract

Heart disease is a term covering any disorder of the heart. Heart diseases have become a major concern to deal with as studies show that the number of deaths due to heart diseases have increased significantly over the past few decades in India it has become the leading cause of death in India. A study shows that from 1990 to 2016 the death rate due to heart diseases have increased around 34% from 155.7 to 209.1 deaths per 1 lakh population in India.

Thus, preventing heart diseases has become more than necessary. Good data-driven systems for predicting heart diseases can improve the entire research and prevention process, making sure that more people can live healthy lives.

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 before 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

2 General Description

2.1 Product Perspective & Problem Statement

The goal of this project is to analyze to predict the probability of heart disease occurrence, based on a combination of features that describes the disease. To achieve the goal, we used a data set that is formed by taking into consideration some of the information of 303 individuals. The problem is based on the given information about each individual we have to calculate that whether that individual will suffer from heart disease or not.

2.2 Tools used

Business Intelligence tools and libraries works such as NumPy, Pandas, Seaborn, Matplotlib, MS-Excel, MS-Power BI, Tableau, Jupyter Notebook and Python Programming Language are used to build the whole framework

3 Design Details

3.1 Functional Architecture

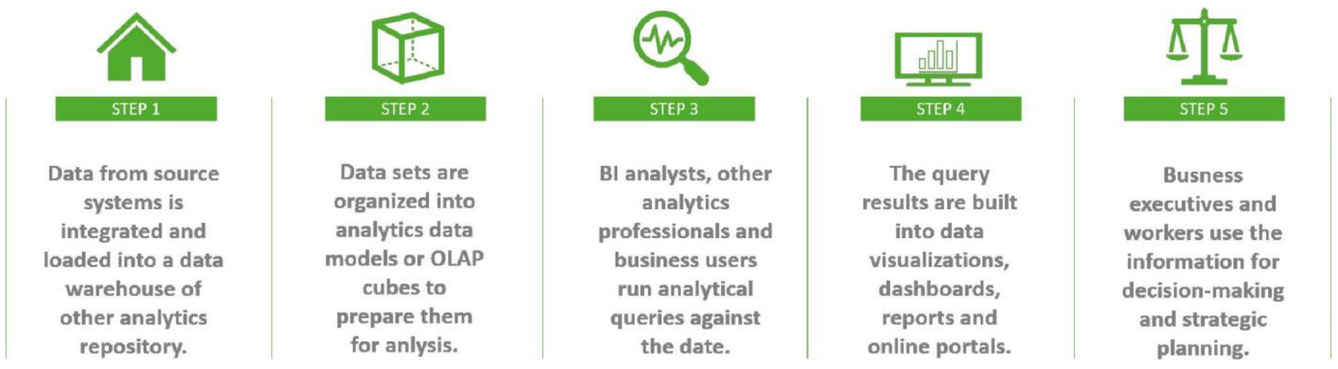
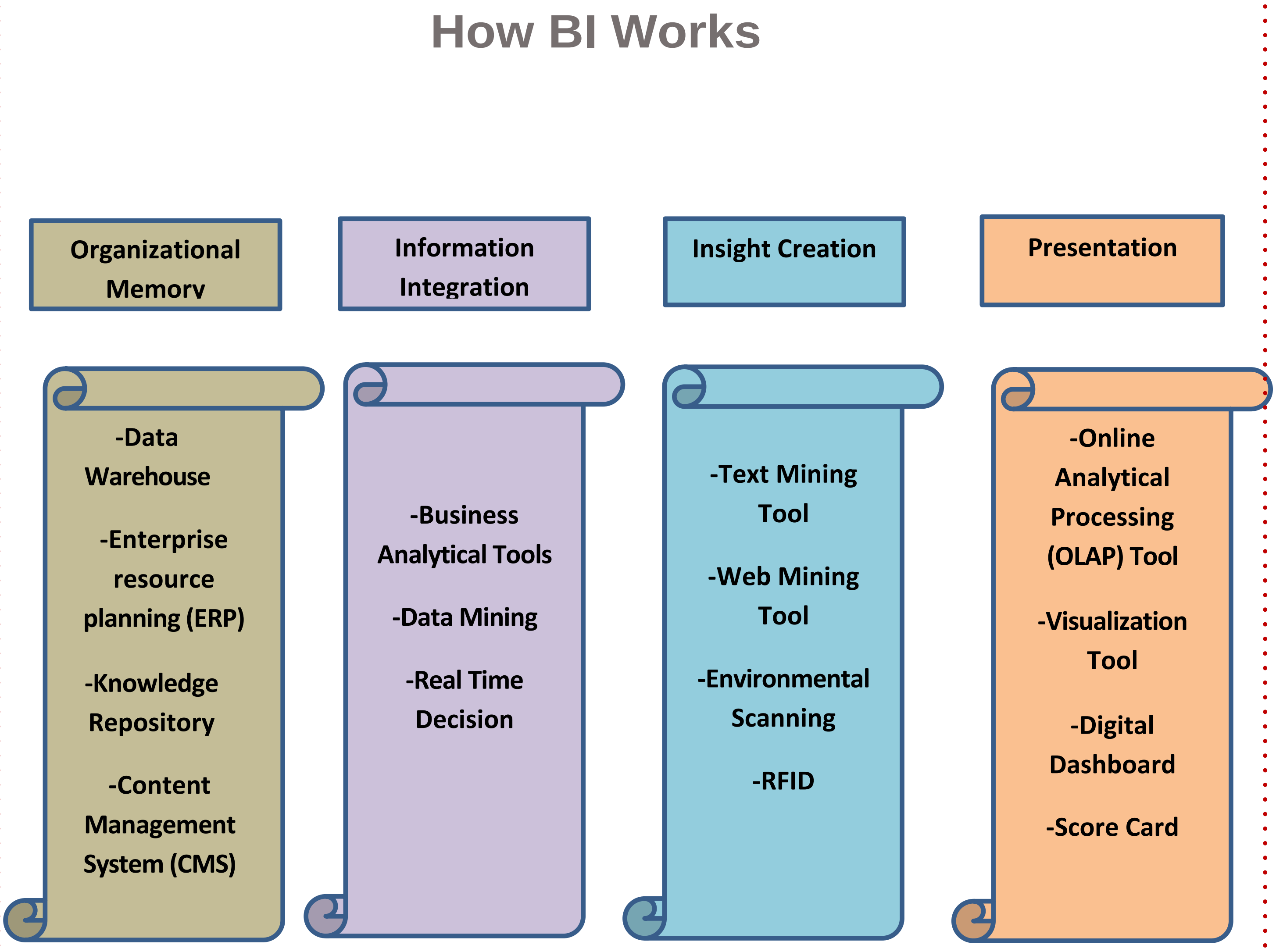

STEP 1
Data from source systems is integrated and loaded into a data warehouse of other analytics repository.
STEP 2
Data sets are organized into analytics data models or OLAP cubes to prepare them for anlysis.
STEP 3
BI analysts, other analytics professionals and business users run analytical queries against the date.
STEP 4
The query results are built into data visualizations, dashboards, reports and online portals.
STEP 5
Busness executives and workers use the information for decision-making and strategic planning.

Figure 1: Functional Architecture of Business Intelligence



3.2 Optimization

1. Your 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

2. Reduce the marks (data points) in your 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.

3. Limit your 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 including 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 data 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)

4. 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.
    - ✓ LODs - Look at the number of unique dimension members in the calculation.
    - ✓ 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 Tableau's 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.

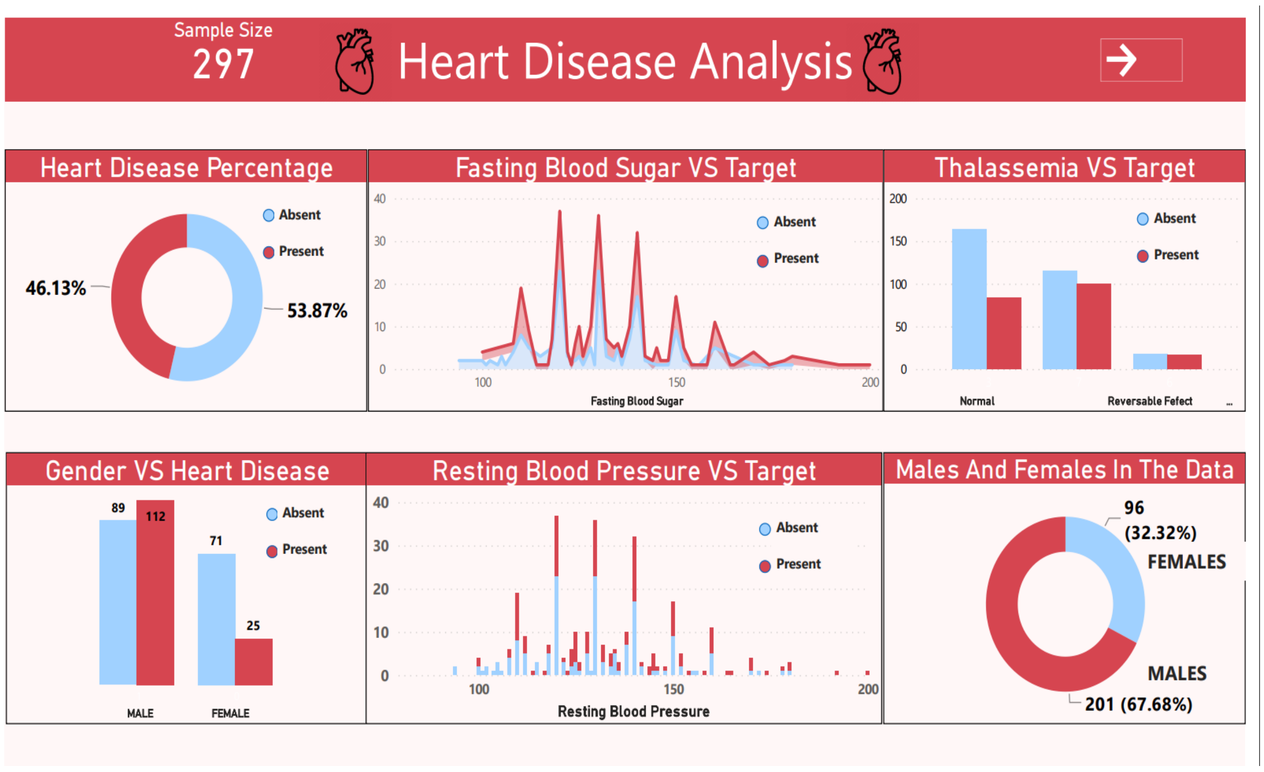
4 Deployment

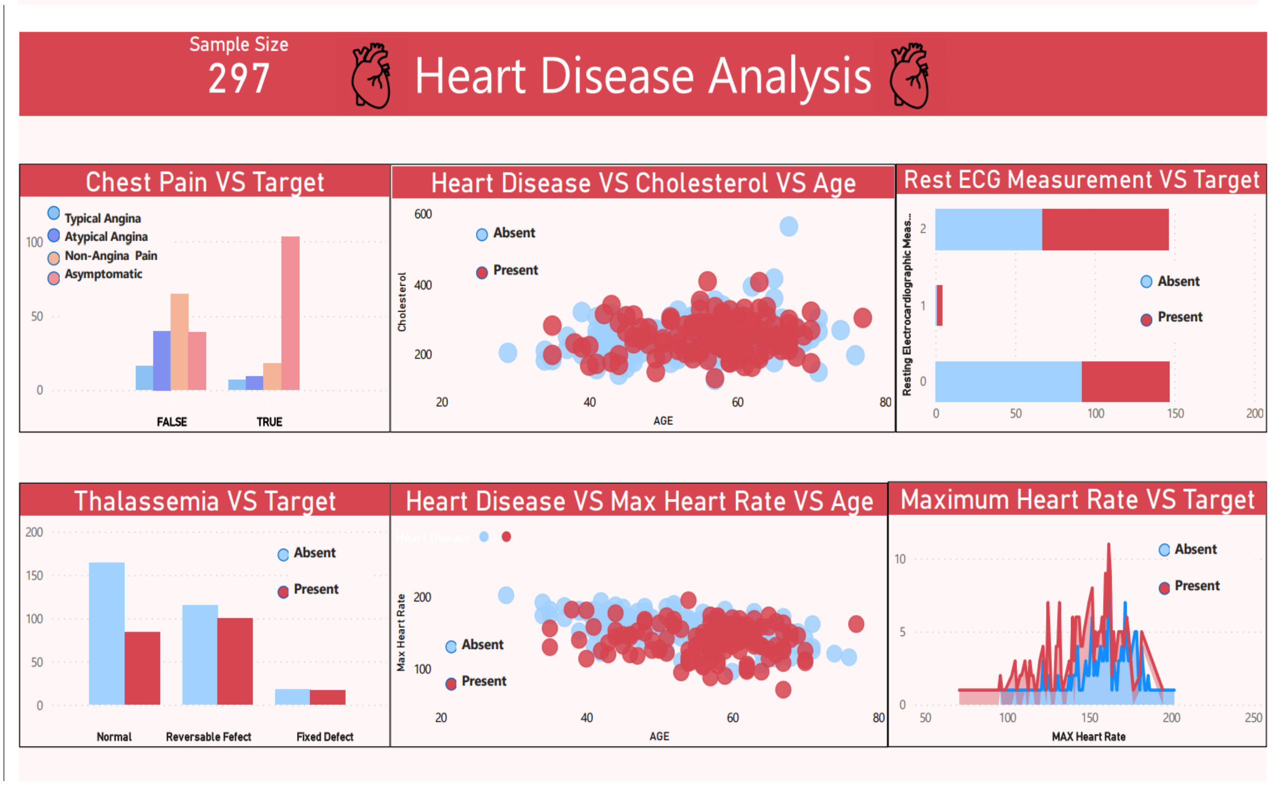
Prioritizing data and analytics couldn't come at a better time. Your company, no matter what size, is already collecting data and most likely analyzing just a portion of it to solve business problems, gain competitive advantages, and drive enterprise transformation. With the explosive growth of enterprise data, database technologies, and the high demand for analytical skills, today's most effective IT organizations have shifted their focus to enabling self-service by deploying and operating Power BI and tableau at scale, as well as organizing, orchestrating, and unifying disparate sources of data for business users and experts alike to author and consume content.

Power BI prioritizes choice in flexibility to fit, rather than dictate, your enterprise architecture. Power BI Desktop and Power BI Service leverage your existing technology

investments and integrate them into your IT infrastructure to provide a self-service, modern analytics platform for your users. With on-premises, cloud, and hosted options, there is a version of Power BI to match your requirements.

Power BI Dashboard


Sample Size
297
Heart Disease Analysis
Heart Disease Percentage
Absent
Present
46.13%
53.87%
Fasting Blood Sugar VS Target
Fasting Blood Sugar
Thalassemia VS Target
Normal
Reversable Fefect
Gender VS Heart Disease
89
112
71
25
MALE
FEMALE
Resting Blood Pressure VS Target
Resting Blood Pressure
Males And Females In The Data
96 (32.32%)
FEMALES
MALES
201 (67.68%)


Sample Size
297
Heart Disease Analysis
Chest Pain VS Target
Typical Angina
Atypical Angina
Non-Angina Pain
Asymptomatic
FALSE
TRUE
Heart Disease VS Cholesterol VS Age
Cholesterol
AGE
Rest ECG Measurement VS Target
Resting Electrocardiographic Meas...
Thalassemia VS Target
Normal
Reversable Fefect
Fixed Defect
Heart Disease VS Max Heart Rate VS Age
Max Heart Rate
Maximum Heart Rate VS Target
MAX Heart Rate
Absent
Present