Phase 5 Documentation

Date	01-11-2023
Team ID	4162
Project Name	Data Warehousing with IBM Cloud Db2 Warehouse

Project Title: Diabetes data warehousing with ETL process using python pandas

Table of Contents

1. Problem Statement

Objective: Developing a diabetes data warehousing system with an ETL process using Python's pandas library to effectively organize and manage diabetes-related data.

2. Project Objective

The project objective is to build a Diabetes Data Warehouse using Python and Pandas with an ETL (Extract, Transform, Load) process. This entails extracting diabetes-related data from various sources, such as patient records, lab results, and lifestyle surveys, using Pandas. The data will be cleansed and transformed to ensure data quality and consistency. It will then be loaded into the data warehouse, creating a structured repository for diabetes-related information. This warehouse will enable data scientists, healthcare professionals, and researchers to analyze and derive actionable insights from the data, facilitating improved patient care and research in the field of diabetes management.

3. Introduction

The Diabetes Data Warehouse Project aims to leverage IBM Db2 Warehouse on IBM Cloud to create a robust data management and analytics platform for diabetes-related data. This documentation outlines the project's objective, design thinking process, and development phases.

4. Literature Survey

1) "Lake Data Warehouse Architecture for Big Data Solutions" Emad Saddad , Ali El-Bastawissy , Hoda M. O. Mokhtar and Maryam Hazman [2020]

The paper introduces the Lake Data Warehouse Architecture, a novel approach that addresses the challenges posed by the abundance of data and the characteristics of big data. This architecture integrates traditional Data Warehouse features with Hadoop and Apache Spark ecosystems to handle large data volumes while ensuring availability and scalability. It aims to enhance business intelligence, data quality, and historical data analysis. The Lake Data Warehouse Architecture supports various data types, including structured, semi-structured, and unstructured data, making it suitable for modern data needs. It also reduces data analysis time and storage costs, making it valuable for data scientists and analytics.

2) "Data Warehousing and Decision Support System Effectiveness Demonstrated in Service Recovery During COVID19 Health Pandemic "Romona M. Harris [2020]

The paper by Romona M. Harris underscores the pivotal role of Data Warehouses (DW) and Decision Support Systems (DSS) in facilitating service recovery during the COVID-19 pandemic. It emphasizes the importance of adapting to service recovery in the face of disruptions and explores various facets of this process. The paper also discusses the significance of understanding customer emotions and satisfaction during recovery efforts. Additionally, it highlights the integration of traditional databases with cloud-based platforms and the need for real-time data analysis.

5. Design Thinking Process

Our project was guided by a design thinking process, which involved iterative problem-solving and a focus on user needs:

1) Empathize:

- Understand the needs of your users and stakeholders. In this case, identify the data requirements and expectations of healthcare professionals, researchers, and other users of the diabetes data warehouse.
- Conduct interviews, surveys, and research to gather insights into what data is necessary and how it should be structured.

2) Define:

- Clearly define the problem statement and project goals. For example, define the specific data sources, data types, and transformations required for your diabetes data warehouse.
- Create a data model and schema design based on the gathered requirements.

3) Ideate:

- Brainstorm and generate creative ideas for implementing the ETL process that will feed data into the MySQL database.
- Consider the optimal way to extract data from source systems, transform it to fit your data model, and load it into the data warehouse.
- Explore various data cleaning and validation techniques that may be necessary.

4) Prototype:

- Develop a prototype ETL pipeline using Python and pandas to validate the design and the data processing steps.
- Create a MySQL database schema that aligns with your data model and supports efficient querying.
- Test the ETL process on a smaller dataset to ensure that data is extracted, transformed, and loaded correctly.

5) Test:

- Conduct thorough testing of the complete ETL process to ensure data integrity, accuracy, and performance.
- Verify that the MySQL database contains the required data and supports various data analysis and reporting needs.
- Gather feedback from users and stakeholders to make necessary adjustments to the ETL process and data warehouse.

6. Development Phases

The project progressed through the following key phases:

6.1 Data Warehouse Structure:

6.1.1 Define Schema:

Schema Definition:

- **Pregnancies (INT):** This column stores the number of times a person has been pregnant. It is of type INT, representing whole numbers.
- **Glucose (INT):** This column stores the plasma glucose concentration after a 2-hour oral glucose tolerance test. It is of type INT.
- **BloodPressure (INT):** This column stores the diastolic blood pressure (mm Hg). It is of type INT.
- **SkinThickness (INT):** This column stores the thickness of the skinfold of the triceps (mm). It is of type INT.
- **Insulin (INT):** This column stores the 2-hour serum insulin (mu U/ml). It is of type INT.
- **BMI (FLOAT):** This column stores the Body Mass Index (weight in kg/(height in m)^2). It is of type FLOAT, allowing decimal values.
- **DiabetesPedigreeFunction (FLOAT):** This column stores a diabetes pedigree function which represents the likelihood of diabetes based on family history. It is of type FLOAT.
- **Age (INT):** This column stores the age of the person (years). It is of type INT.
- **Outcome (BINARY):** This column stores binary values representing the outcome of whether a person has diabetes or not. The exact representation of these binary values (e.g., 0 and 1) would need to be defined based on the specific context of the dataset. It is of type BINARY.

6.1.2 Structure of the Data Warehouse Tables:

CREATE TABLE diabetes_data:

This line initiates the creation of a new table named diabetes_data in the database.

SQL QUERY:

```
CREATE TABLE diabetes_data
(
Pregnancies INT,
Glucose INT,
BloodPressure INT,
SkinThickness INT,
Insulin INT,
BMI float,
DiabetesPedigreeFunction float,
Age INT,
Outcome BINARY
);
SELECT * FROM diabetes_data;
```

SELECT: This statement is used to select data from the table. which means it selects all columns from the specified table.

FROM diabetes_data: This part of the statement specifies the source table from which data is being selected, which is diabetes_data.

The purpose of the script is to create a table that can store data related to diabetes, and the SELECT * FROM diabetes_data statement is used to retrieve all the records (rows) from this table.

6.2 Data Integration:

- The objective of this project is to classify whether someone has diabetes or not.
- Dataset consists of several Medical Variables (Independent) and one Outcome Variable (Dependent)
- The independent variables in this data set are: -'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI' ,'DiabetesPedigreeFunction', 'Age'
- The outcome variable value is either 1 or 0 indicating whether a person has diabetes (1) or not (0).

Dataset: https://www.kaggle.com/code/mvanshika/diabetes-prediction.

6.3 ETL Process

The ETL (Extract, Transform, Load) process is a fundamental data integration process used to collect, clean, transform, and load data from various sources into a target data repository, such as a database, data warehouse, or data lake.

6.3.1 Extract:

Data extraction is the first step in the ETL (Extract, Transform, Load) process, where data is collected or retrieved from one or more source systems for further processing. In your specific project of loading a diabetes CSV file into a MySQL database, data extraction involves obtaining the diabetes data from the CSV file and loading it into a Python environment for further transformation and loading into the database.

```python
import pandas as pd
df = pd.read_csv('diabetes_dataset.csv')
```

This code reads the CSV file and stores the data in the df DataFrame.

To use Python's pandas library to load the data from the CSV file. The pd.read_csv() function is a common method for reading data from CSV files into a Pandas DataFrame, which is a tabular data structure. The CSV file is typically located in your local directory or at a specified file path.

6.3.2 Transform:

The data transformation step in the ETL (Extract, Transform, Load) process is crucial for preparing the raw data extracted from the source (in this case, a diabetes CSV file) for loading into a MySQL database. Transformation involves cleaning, structuring, and enriching the data to ensure it is in the right format and quality for its intended use.

- Dropping duplicate values
- Checking NULL values
- Checking for 0 value and replacing it :- It isn't medically possible for some data record to have 0 value such as Blood Pressure or Glucose levels. Hence we replace them with the mean value of that particular column.

```python
df.info()
df.isnull().sum()
```

```
In [3]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
In [4]: df.isnull().sum()
```

```
Out[4]: Pregnancies                 0
        Glucose                     0
        BloodPressure               0
        SkinThickness               0
        Insulin                     0
        BMI                         0
        DiabetesPedigreeFunction    0
        Age                         0
        Outcome                     0
        dtype: int64
```

```
print(df[df['BloodPressure']==0].shape[0])
print(df[df['Glucose']==0].shape[0])
print(df[df['SkinThickness']==0].shape[0])
print(df[df['Insulin']==0].shape[0])
print(df[df['BMI']==0].shape[0])
df=df.drop_duplicates()
df.describe()
```

```
In [5]: print(df[df['BloodPressure']==0].shape[0])
        print(df[df['Glucose']==0].shape[0])
        print(df[df['SkinThickness']==0].shape[0])
        print(df[df['Insulin']==0].shape[0])
        print(df[df['BMI']==0].shape[0])
```

```
35
5
227
374
11
```

```
In [6]: df=df.drop_duplicates()
```

```
In [7]: df.describe()
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

6.3.3 Load:

In the ETL (Extract, Transform, Load) process, the "Load" step is the final phase where the transformed data is loaded into the target destination, which is typically a database. In the project of extracting and transforming diabetes data from a CSV file, this step involves loading the cleaned and structured data into a MySQL database. Here's a detailed explanation of the data loading step:

To load data into a MySQL database, First need to establish a connection to the MySQL server. To require necessary credentials to access the database server.

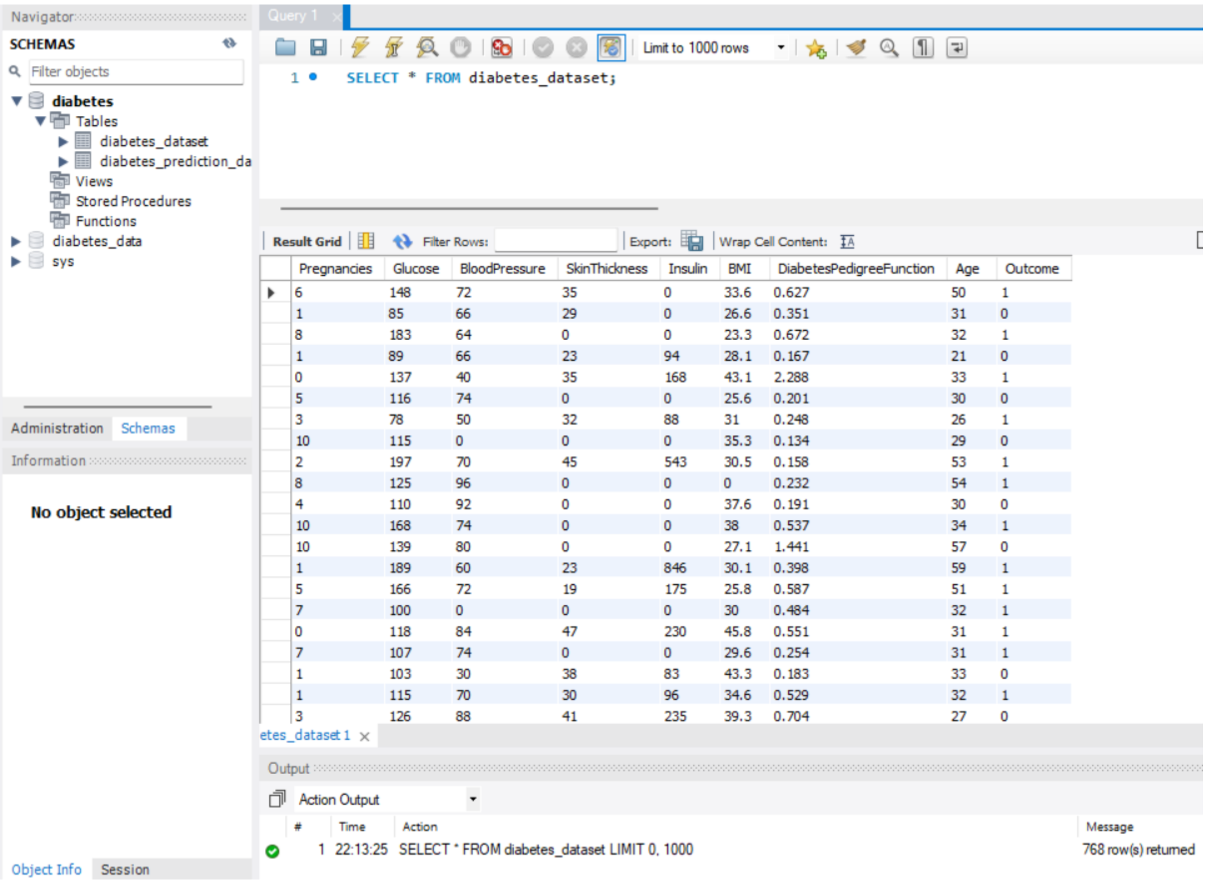
```
import mysql.connector
conn = mysql.connector.connect(
    host='local',
    user='root',
    password='pass_word',
    database='diabetes_data'
)

cursor = conn.cursor()
```

To define the structure of the table in the MySQL database where the data will be stored. This structure should match the schema of the transformed data.

```
create_table_query = "CREATE TABLE diabetes_pred (Pregnancies INT, Glucose INT, BloodPressure INT, SkinThickness INT, Insulin INT, BMI float, DiabetesPedigreeFunction float, Age INT, Outcome BINARY);"

cursor.execute(create_table_query)

for index, row in df.iterrows():
        cursor.execute("INSERT INTO diabetes_pred (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age,
```

```
Outcome) VALUES (%s, %s, %s, %s,%s, %s,%s, %s,%s);"
            ,(row['Pregnancies'], row['Glucose'], row['BloodPressure'],
row['SkinThickness'], row['Insulin'], row['BMI'],
row['DiabetesPedigreeFunction'], row['Age'], row['Outcome']))

conn.commit()
conn.close()
```

The "Load" step completes the ETL process by moving the transformed data from your Python environment into a MySQL database, making it accessible for querying and analysis within the database system. This step ensures that the data is structured, organized, and stored in a way that allows for efficient retrieval and analysis.



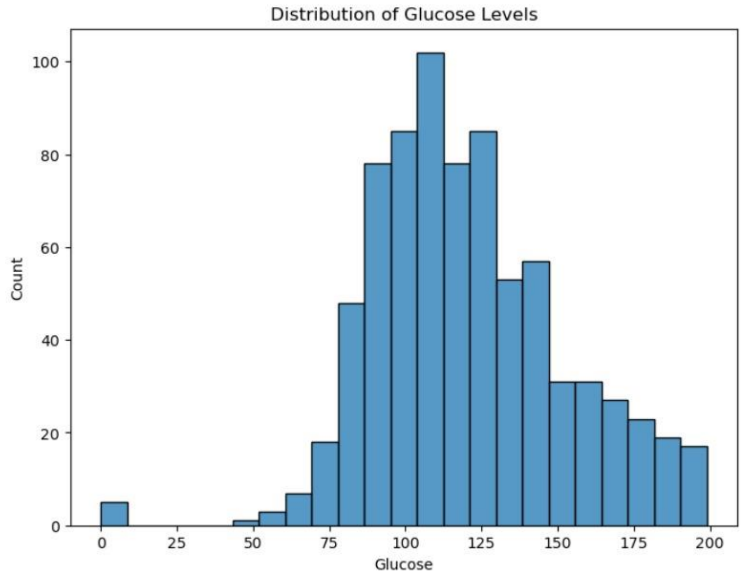
6.4 Data Exploration

Data exploration is a crucial step in any data project, including building a diabetes data warehouse. It involves examining and understanding the data you have collected or will collect. This process helps identify patterns, trends, anomalies, and insights within the data. In this project, we can use Python libraries like pandas, matplotlib, and seaborn to perform data exploration.

Visualize your data to gain insights. Use libraries like matplotlib and seaborn to create various plots, such as histograms, box plots, and scatter plots to understand data distribution and relationships.

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(8, 6))
sns.histplot(df  ['glucose_level'])
plt.title('Distribution of Glucose Levels')
plt.show()
```

```
In [15]: import matplotlib.pyplot as plt
         import seaborn as sns

In [18]: plt.figure(figsize=(8, 6))
         sns.histplot(df['Glucose'])
         plt.title('Distribution of Glucose Levels')

Out[18]: Text(0.5, 1.0, 'Distribution of Glucose Levels')
```



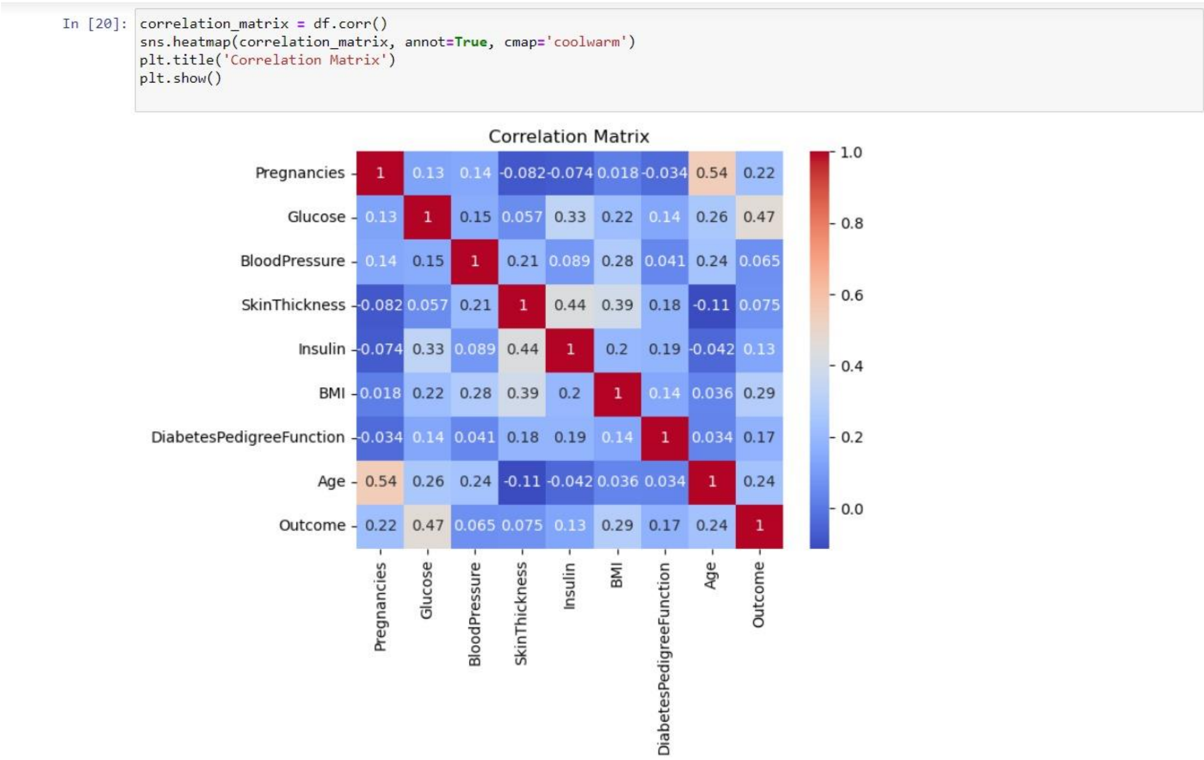
Calculate and visualize the correlations between variables to understand how they are related. This is important for feature selection and model building in the future.

correlation_matrix = df.corr()

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')

plt.title('Correlation Matrix')

plt.show()



7. Actionable Insights

- A diabetes data warehouse plays a pivotal role in enabling data architects to deliver actionable insights.
- By consolidating diverse data sources into a centralized repository, it provides a unified view of patient information, treatments, and outcomes.
- Data architects can design sophisticated ETL processes to clean, transform,

and integrate data efficiently.

- Structuring data in a way that aligns with healthcare standards allows for complex querying and analysis.
- With a well-organized data warehouse, data architects can leverage advanced analytics tools and algorithms to identify trends, patterns, and correlations within the diabetes data.
- These insights can lead to actionable conclusions, such as optimizing treatment protocols, predicting patient outcomes, and identifying high-risk individuals.
- Moreover, data warehouses facilitate historical analysis, enabling healthcare providers to track disease progression over time and evaluate the effectiveness of interventions.

8. Conclusion

The Diabetes Data Warehouse project, driven by design thinking, ETL processes, and data exploration, has successfully established a robust foundation for diabetes care and research. By centralizing and structuring diverse data sources, it empowers healthcare professionals and researchers to derive actionable insights, driving personalized treatment strategies and enhancing patient outcomes. The project showcases the power of data-driven healthcare solutions in revolutionizing diabetes management.