Department of Computer Science and Engineering (Data Science)

Subject: Applied Data Science (DJ19DSL703)

Experiment -5

(Data Modelling)

Name: Rishabh Patil

SAP ID: 60009200056

Batch: D12

Aim: To complete Data Modelling using appropriate tool.

Theory:

Data modelling is a crucial aspect of data science. It involves creating a conceptual representation of data to help data scientists and analysts understand and work with data effectively. There are several types of data modelling, and the choice of modelling technique depends on the specific requirements of the project.

Data modelling is the process of creating a visual representation of either a whole information system or parts of it to communicate connections between data points and structures. The goal is to illustrate the types of data used and stored within the system, the relationships among these data types, the ways the data can be grouped and organized and its formats and attributes.

Data models are built around business needs. Rules and requirements are defined upfront through feedback from business stakeholders so they can be incorporated into the design of a new system or adapted in the iteration of an existing one.

Data can be modelled at various levels of abstraction. The process begins by collecting information about business requirements from stakeholders and end users. These business rules are then translated into data structures to formulate a concrete database design. A data model can be compared to a roadmap, an architect's blueprint or any formal diagram that facilitates a deeper understanding of what is being designed.

Data modelling employs standardized schemas and formal techniques. This provides a common, consistent, and predictable way of defining and managing data resources across an organization, or even beyond.

Types of data models

Data models can generally be divided into three categories, which vary according to their degree of abstraction. Like any design process, database and information system design begins at a high level of abstraction and becomes increasingly more concrete and specific. Data models can generally be divided into three categories, which vary according to their degree of abstraction. The process will start with a conceptual model, progress to a logical model and conclude with a physical model.

1. Conceptual data models

They are also referred to as domain models and offer a big-picture view of what the system will contain, how it will be organized, and which business rules are involved. Conceptual models are usually created as part of the process of gathering initial project requirements. Typically, they include entity classes (defining the types of things that are important for the business to represent in the data model), their characteristics and constraints, the relationships between them and relevant security and data integrity requirements. Any notation is typically simple.

2. Logical data models

They are less abstract and provide greater detail about the concepts and relationships in the domain under consideration. One of several formal data modeling notation systems is followed. These indicate data attributes, such as data types and their corresponding lengths, and show the relationships among entities. Logical data models don’t specify any technical system requirements. This stage is frequently omitted in agile or DevOps practices. Logical data models can be useful in highly procedural implementation environments, or for projects that are data-oriented by nature, such as data warehouse design or reporting system development.

3. Physical data models

They provide a schema for how the data will be physically stored within a database. As such, they’re the least abstract of all. They offer a finalized design that can be implemented as a relational database, including associative tables that illustrate the relationships among entities as well as the primary keys and foreign keys that will be used to maintain those relationships. Physical data models can include database management system (DBMS)-specific properties, including performance tuning.

Data modelling process:

1. **Identify the entities.** The process of data modeling begins with the identification of the things, events or concepts that are represented in the data set that is to be modeled.Each entity should be cohesive and logically discrete from all others.
2. **Identify key properties of each entity.** Each entity type can be differentiated from all others because it has one or more unique properties, called attributes. For instance, an entity called “customer” might possess such attributes as a first name, last name, telephone number and salutation, while an entity called “address” might include a street name and number, a city, state, country and zip code.
3. **Identify relationships among entities.** The earliest draft of a data model will specify the nature of the relationships each entity has with the others. In the above example, each customer “lives at” an address. If that model were expanded to include an entity

called "orders," each order would be shipped to and billed to an address as well. These relationships are usually documented via unified modeling language (UML).

4. **Map attributes to entities completely.** This will ensure the model reflects how the business will use the data. Several formal data modeling patterns are in widespread use. Object-oriented developers often apply analysis patterns or design patterns, while stakeholders from other business domains may turn to other patterns.
5. **Assign keys as needed, and decide on a degree of normalization that balances the need to reduce redundancy with performance requirements.** Normalization is a technique for organizing data models (and the databases they represent) in which numerical identifiers, called keys, are assigned to groups of data to represent relationships between them without repeating the data. For instance, if customers are each assigned a key, that key can be linked to both their address and their order history without having to repeat this information in the table of customer names. Normalization tends to reduce the amount of storage space a database will require, but it can at cost to query performance.
6. **Finalize and validate the data model.** Data modeling is an iterative process that should be repeated and refined as business needs change.

Lab Assignment:

Use [dbdiagram.io (online tool)] for creating Entity relationship diagram.

We then create a database which will reflect the same data model architecture.

ETL stands for "Extract, Transform, Load," and it refers to a process used in data integration and data warehousing. ETL is a crucial part of managing and preparing data for analysis or reporting in business intelligence and data analytics. Here's what each step of the ETL process entails:

1) Extract: In the first step, data is extracted from various sources. These sources can include databases, flat files, web services, APIs, logs, and more. The goal is to gather data from disparate sources and consolidate it into a unified location for further processing.

2) Transform: After data extraction, the data is transformed. This involves cleaning, structuring, and enriching the data to make it suitable for analysis. Data transformation may include tasks such as filtering out irrelevant information, handling missing values, standardizing data formats, aggregating data, and performing calculations. It's during this step that noisy data can be cleaned up, and the data is often transformed into a format that is suitable for the target system or analytics tools.

3) Load: The final step involves loading the transformed data into a target data repository or data warehouse. This repository is optimized for querying and reporting, making it easier to access and analyse the data. The data can be stored in a relational database, a data lake, or another suitable storage solution. It's important to maintain data integrity during this process and ensure that the data remains consistent and up-to-date.

ETL processes are typically implemented using ETL tools and platforms, which automate much of the workflow. These tools allow data engineers and analysts to define data extraction, transformation, and loading tasks in a visually intuitive manner. ETL is a critical step in data management, as it enables organizations to integrate and prepare data from different sources for business intelligence, reporting, and analytics, ensuring that the data is accurate, consistent, and ready for decision-making.

For the next steps make use of one of the following tools which can help in ETL (Extract, Transform, Load) operation.

- **Traditional Coding (Python)**
- **Low Code – No Code Tools (PyCaret, H20 AutoML, Altair Rapidminer)**

Create an ETL pipeline using Python or any of the low code no code tools mentioned above. The steps will be –

- Prepare the steps to clean the data as per the data definition decided during the data modelling step.
- Create an ETL pipeline to incorporate the pre-processing steps.
- Load the data in the database tables.

Use of this: (for Incremental Data)

https://www.kaggle.com/datasets/karkavelrajaj/amazon-sales-dataset

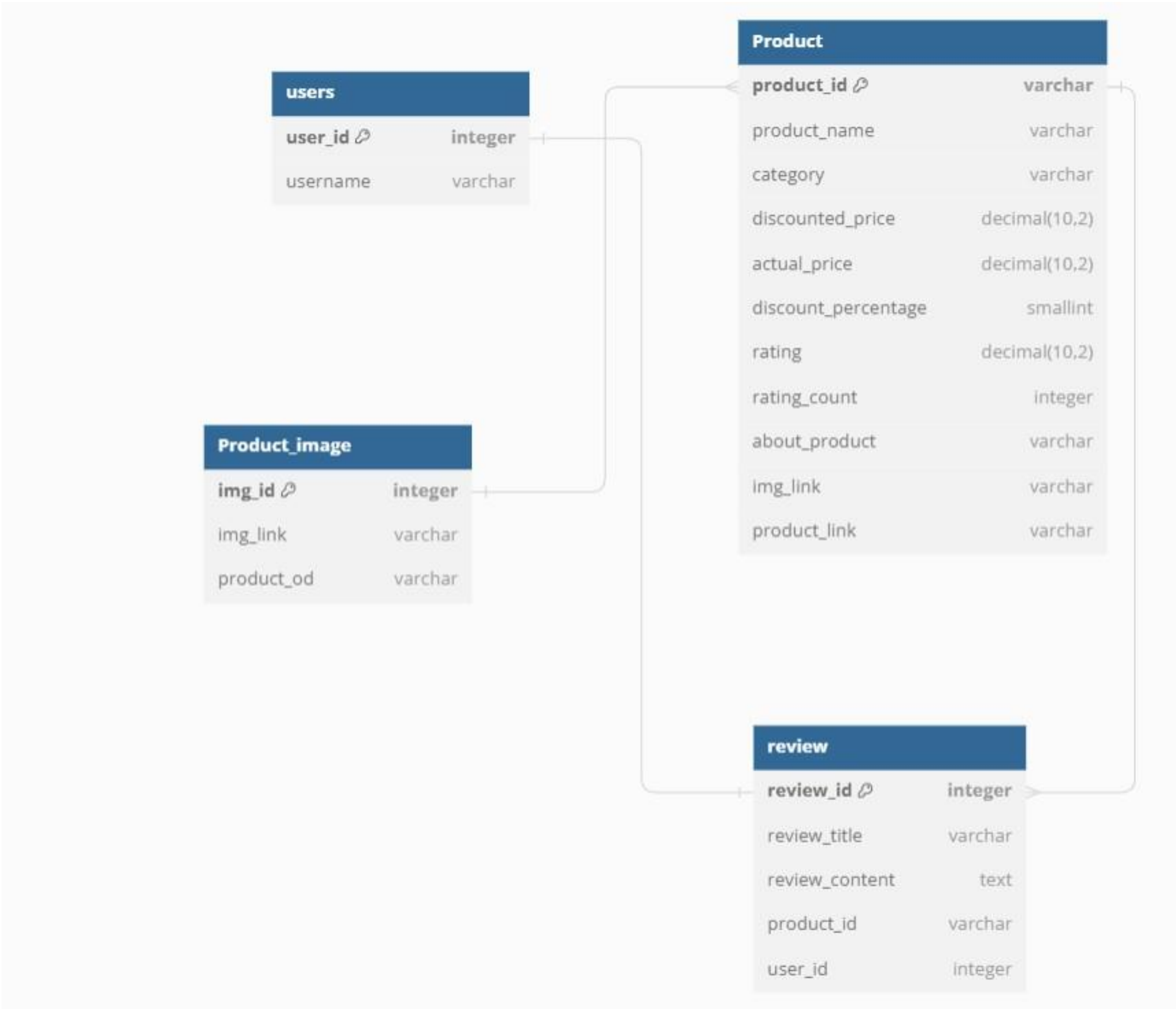
To load 60% of the data using the create ETL pipeline. Then we can upload the next 40% data in 4 batches of 10% data being comprised in each batch.

Also Identify noise in the dataset along with the preprocessing measures taken to the remove the noise from the data set.

1. Creating ER Diagram

```
// Use DBML to define your database structure
// Docs: https://dbml.dbdiagram.io/docs

Table Product {
  product_id varchar [primary key]
  product_name varchar
  category varchar
  discounted_price decimal(10,2)
  actual_price decimal(10,2)
  discount_percentage smallint
  rating decimal(10,2)
  rating_count integer
  about_product varchar
  img_link varchar
  product_link varchar
}
```

```
Table users {
  user_id integer [primary key]
  username varchar
}

Table review {
  review_id integer [primary key]
  review_title varchar
  review_content text
  product_id varchar
  user_id integer
}

Table Product_image {
  img_id integer [primary key]
  img_link varchar
  product_od varchar
}

Ref: review.review_id > Product.product_id // many-to-one

Ref: Product_image.img_id < Product.product_id

Ref: review.review_id - users.user_id
```



2. ETL and Data Storage on PostGRE Sql

```
import psycopg2
import pandas as pd
import numpy as np
from sqlalchemy import create_engine

# Define your database connection parameters for psycopg2
db_params = {
    "host": "localhost",
    "database": "demo",
    "user": "postgres",
    "password": "123"
}

# Load your dataset (assuming you have a CSV file)
data = pd.read_csv("amazon.csv")
```

```
# Data Cleaning and Noise Removal
# In this example, we'll replace missing values with NaN and remove duplicate
rows.
data = data.fillna(np.nan)  # Replace missing values with NaN
data = data.drop_duplicates()  # Remove duplicates

# Split the data into 60% and 40%
total_records = len(data)
split_point = int(0.6 * total_records)
data_60_percent = data[:split_point]  # First 60% of the data

# Split the remaining 40% into 4 equal batches
batch_size = int(0.1 * total_records)
batches = [data[i:i + batch_size] for i in range(split_point, total_records,
batch_size)]

try:
    # Create a connection to the PostgreSQL database using psycopg2
    conn = psycopg2.connect(**db_params)
    cursor = conn.cursor()

    # Load the first 60% of data into the database using SQLAlchemy engine
    postgresql_engine =
create_engine("postgresql://postgres:123@localhost/demo")
    data_60_percent.to_sql("adslab5", postgresql_engine, if_exists='replace',
index=False)

    # Load the subsequent 4 batches into the database using SQLAlchemy engine
    for batch in batches:
        batch.to_sql("adslab5", postgresql_engine, if_exists='append',
index=False)

    # Commit the changes to the database
    conn.commit()

except Exception as e:
    print(f"Error: {e}")

finally:
    if conn:
        cursor.close()
        conn.close()
```

Department of Computer Science and Engineering (Data Science)