SQL Assignment: Creating a Library management system database

Introduction

This report offers justification for the design decisions made in structuring a Library Management System database, along with an overview of the ethical considerations integrated during its development. The database simulates a real-world system for managing books, memberships, and the condition of books within a library or bookstore context. It aims to achieve two objectives: to be both realistic and ethically sound, ensuring compliance with principles of privacy and integrity.

The report is divided into three sections: a justification for the tables and attributes, an examination of ethical considerations, and a conclusion summarizing the key points discussed.

How the data was generated

The data for this database was generated using Python. The process involved creating random data with the NumPy, Matplotlib.pyplot and Pandas library. Random values were assigned to the following columns: Book_Price, Published_Year, Number_of_Pages, Membership_Join_Date, and Membership_Expiration_Date. For categorical data such as Genre, Author, Publisher, and Membership_Type, predefined lists with realistic values were created, from which random selections were made.

To ensure controlled distributions membership types and book conditions were assigned probabilities to reflect real-world frequency distributions, details such as having more members in the “Bronze” category than in “Platinum” were made.

Missing data was deliberately introduced in the Book_Price and Number_of_Pages columns to simulate real-world challenges where datasets often contain incomplete records. Foreign keys and composite keys were used to establish relationships between tables, ensuring normalization and data consistency. The names of authors and publishers were selected from realistic yet entirely fictional lists to create a professional and believable dataset.

Screenshots of Books table

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Number of rows for Books Table
n_books = 1000

# Nominal data
genres = ['Romance', 'Fantasy', 'Mystery', 'Science Fiction', 'Horror']

# Author names
authors = [
    "James Smith", "Mary Johnson", "Robert Brown", "Patricia Garcia", "Michael Martinez",
    "Linda Robinson", "William Clark", "Elizabeth Lewis", "David Walker", "Barbara Hall",
    "Richard Young", "Susan Allen", "Joseph Hernandez", "Sarah King", "Thomas Wright",
    "Karen Lopez", "Charles Hill", "Nancy Scott", "Christopher Green", "Betty Adams",
    "Daniel Baker", "Margaret Gonzalez", "Paul Nelson", "Sandra Carter", "Mark Mitchell",
    "Lisa Perez", "Donald Roberts", "Emily Turner", "George Phillips", "Jessica Parker",
    "Kenneth Evans", "Sharon Edwards", "Steven Collins", "Laura Stewart", "Andrew Morris",
]
```

```python
# Books DataFrame
books_df = pd.DataFrame({
    'Book_ID': range(1, n_books + 1),    # Unique ID
    'Genre': genre_data,
    'Author': author_data,
    'Publisher': publisher_data,
    'Book_Condition': condition_data,
    'Published_Year': published_year_data,
    'Book_Price': book_price_data,
    'Number_of_Pages': page_data
})

print(books_df.head())
print(f"Total rows in Books table: {len(books_df)}")
```

	Book_ID	Genre	Author	Publisher	Book_Condition	Published_Year	Book_Price	Number_of_Pages
0	1	Science Fiction	Sandra Carter	SAGE Publications	Good	1902	15	394
1	2	Science Fiction	Charles Hill	Taylor & Francis	Fair	1937	48	481
2	3	Science Fiction	Laura Stewart	Macmillan Publishers	Excellent	1821	12	425
3	4	Science Fiction	Susan Allen	HarperCollins	Good	2008	23	266
4	5	Horror	Patricia Garcia	Scholastic Corporation	Good	2001	14	403

Total rows in Books table: 1000

Screenshots of Membership Table

```python
# Number of rows for Membership Table
n_members = 50

# Ordinal data
membership_types = ['Bronze', 'Silver', 'Gold', 'Platinum']

# Generate Membership Data
membership_data = np.random.choice(membership_types, n_members, p=[0.4, 0.3, 0.2, 0.1])

# Interval data for dates
join_dates = pd.date_range('2010-01-01', '2023-01-01', periods=n_members)
exp_dates = join_dates + pd.DateOffset(years=1)

# Membership DataFrame
membership_df = pd.DataFrame({
    'Member_ID': range(1, n_members + 1),
    'Membership_Type': membership_data,
    'Join_Date': join_dates,
    'Expiration_Date': exp_dates
})

print(membership_df.head())
print(f"Total rows in Membership table: {len(membership_df)}")
```

```
   Member_ID Membership_Type                   Join_Date  \
0          1        Platinum 2010-01-01 00:00:00.000000000
1          2          Silver 2010-04-07 21:33:03.673469388
2          3          Bronze 2010-07-13 19:06:07.346938776
3          4            Gold 2010-10-18 16:39:11.020408164
4          5          Bronze 2011-01-23 14:12:14.693877552

               Expiration_Date
0 2011-01-01 00:00:00.000000000
1 2011-04-07 21:33:03.673469388
2 2011-07-13 19:06:07.346938776
3 2011-10-18 16:39:11.020408164
4 2012-01-23 14:12:14.693877552
Total rows in Membership table: 50
```

Screenshots of Book Condition Table

```python
n_conditions = 50

condition_data = np.random.choice(book_conditions, n_conditions, p=[0.1, 0.2, 0.4, 0.3])

last_updated = pd.date_range('2020-01-01', '2023-01-01', periods=n_conditions)

book_ids = np.random.randint(1, n_books + 1, n_conditions)

condition_df = pd.DataFrame({
    'Condition_ID': range(1, n_conditions + 1),
    'Book_ID': book_ids,
    'Book_Condition': condition_data,
    'Last_Updated': last_updated
})

print(condition_df.head())
print(f"Total rows in Book Condition table: {len(condition_df)}")
```

```
   Condition_ID  Book_ID Book_Condition                 Last_Updated
0             1      557           Good 2020-01-01 00:00:00.000000000
1             2      501           Good 2020-01-23 08:48:58.775510204
2             3      322      Excellent 2020-02-14 17:37:57.551020408
3             4      160      Excellent 2020-03-08 02:26:56.326530612
4             5      443           Fair 2020-03-30 11:15:55.102040816
Total rows in Book Condition table: 50
```

Database Schema

The database consists of three main tables: Books, Library Memberships, and Book Condition. Below is the schema:

Books Table:

- **Attributes**:
    - Book_ID (Primary Key, Unique)
    - Genre (Nominal)
    - Author (Nominal)
    - Publisher (Nominal)
    - Book_Condition (Ordinal: Poor, Fair, Good, Excellent)

    - Published_Year (Interval)
    - Book_Price (Ratio)
    - Number_of_Pages (Ratio)
- **Constraints**:
    - Book_ID must be unique
    - Published_Year must fall between 1800 and 2023

Memberships Table:

- **Attributes**:
    - Member_ID (Primary Key, Unique)
    - Membership_Type (Ordinal: Bronze, Silver, Gold, Platinum)
    - Join_Date (Interval)
    - Expiration_Date (Interval)
- **Constraints**:
    - Membership types are limited to the predefined tiers
    - Expiration_Date is calculated as one year after Join_Date

Book Condition Table:

- **Attributes**:
    - Condition_ID (Primary Key, Unique)
    - Book_ID (Foreign Key referencing Books.Book_ID)
    - Book_Condition (Ordinal: Poor, Fair, Good, Excellent)
    - Last_Updated (Interval)
- **Constraints**:
    - Book_ID must exist in the Books table
    - Last_Updated must be a valid date

Justification for Separate Tables

The Books Table is designed to systematically store detailed information about all books, effectively preventing data duplication. The Membership Table manages individual member records, each with distinct details, which streamlines the handling of member information and keeps the Books Table uncluttered. Meanwhile, the Book Condition Table tracks how the condition of books changes over time, facilitating audits, pricing based on their condition, and efficient inventory management. In terms of ethics and data privacy, the dataset safeguards

privacy by omitting any personally identifiable information (PII) and anonymising membership details, including Membership Type and Join Date. The names of authors and publishers are fictional but realistic, ensuring there are no unintended references to actual individuals or organizations. Prices and publication years are presented in a realistic yet generalized manner to avoid misrepresentation. The use of randomized data generation minimises biases and promotes fairness across various genres, authors, and publishers, reflecting real-world trends in the distribution of membership types and book conditions (for example, having more "Good" condition books than "Poor"). Additionally, the inclusion of missing data is a deliberate choice to replicate real-world situations and provide opportunities to practice handling incomplete datasets. The employment of foreign and composite keys further ensures the database is consistent and logically structured, which significantly reduces the chances of errors or inconsistencies.

Conclusion

In conclusion the database was designed with a focus on realism, functionality, and ethical considerations. The data was generated using Python, leveraging randomization techniques and realistic constraints. The schema consists of well-structured tables connected through foreign keys, enabling efficient and meaningful data management. Ethical principles, including data privacy, unbiased representation, and transparency, were prioritized throughout the design process.