Creation of Employee Management System Database

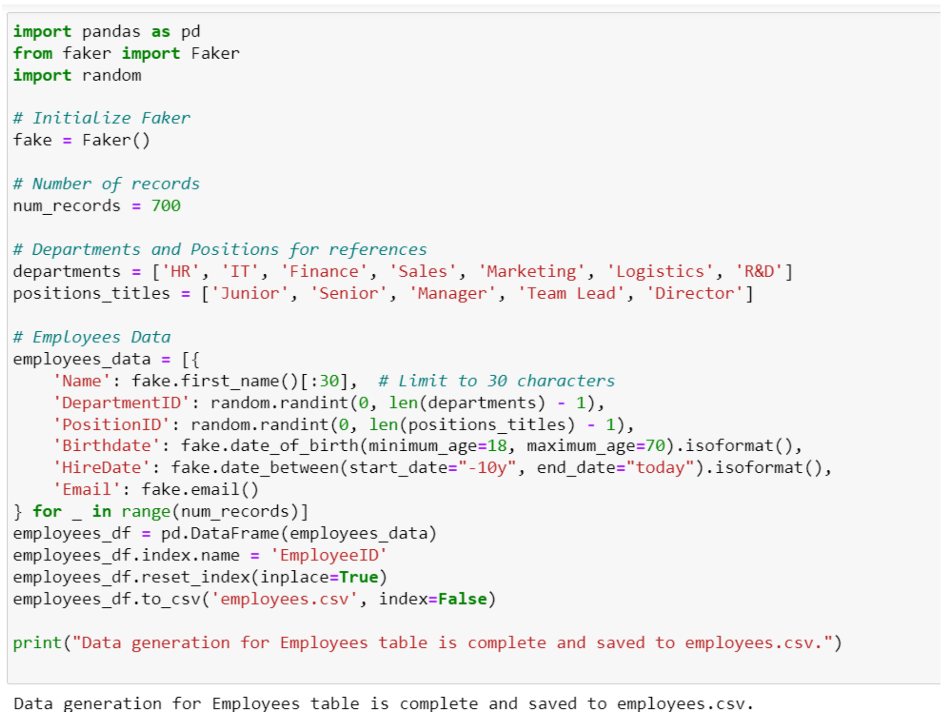
Name	**Muhammad Awais**
Student ID	21084741
Course	Data Mining and Discovery
Assignment	SQL Assignment
Submission Date	12-11-2023

A. Data Generation

I selected the topic of Employee Management System for which I decided to make the database. The data for our Employee Management System was generated using the Python programming language, with the help of the pandas library for data manipulation and the Faker library for random but realistic data generation. Specifically:

Employees Table

Randomized data was generated for employees, including their name, email, department, and position IDs, birthdate, and hire date.



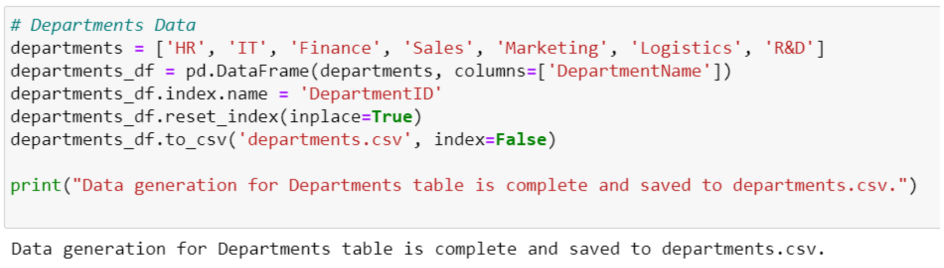
```python
import pandas as pd
from faker import Faker
import random

# Initialize Faker
fake = Faker()

# Number of records
num_records = 700

# Departments and Positions for references
departments = ['HR', 'IT', 'Finance', 'Sales', 'Marketing', 'Logistics', 'R&D']
positions_titles = ['Junior', 'Senior', 'Manager', 'Team Lead', 'Director']

# Employees Data
employees_data = [{
    'Name': fake.first_name()[:30],  # Limit to 30 characters
    'DepartmentID': random.randint(0, len(departments) - 1),
    'PositionID': random.randint(0, len(positions_titles) - 1),
    'Birthdate': fake.date_of_birth(minimum_age=18, maximum_age=70).isoformat(),
    'HireDate': fake.date_between(start_date="-10y", end_date="today").isoformat(),
    'Email': fake.email()
} for _ in range(num_records)]
employees_df = pd.DataFrame(employees_data)
employees_df.index.name = 'EmployeeID'
employees_df.reset_index(inplace=True)
employees_df.to_csv('employees.csv', index=False)

print("Data generation for Employees table is complete and saved to employees.csv.")
```

```
Data generation for Employees table is complete and saved to employees.csv.
```

Figure 1: Code to Generate fake data for Employees Table

Departments Table

Seven departments were predefined, such as HR, IT, Finance, Sales, etc.



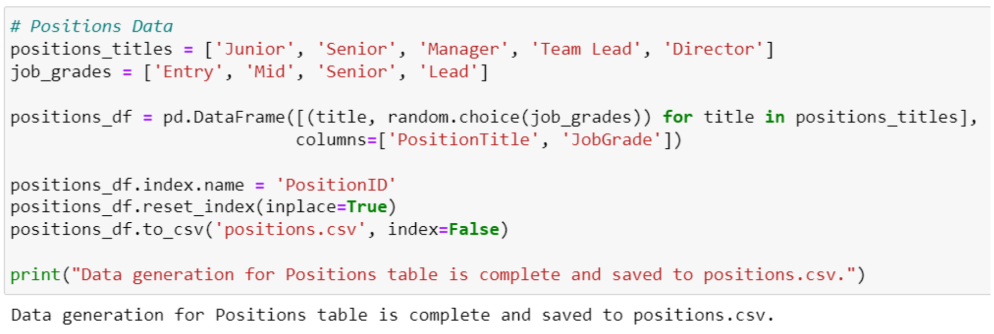
```
# Departments Data
departments = ['HR', 'IT', 'Finance', 'Sales', 'Marketing', 'Logistics', 'R&D']
departments_df = pd.DataFrame(departments, columns=['DepartmentName'])
departments_df.index.name = 'DepartmentID'
departments_df.reset_index(inplace=True)
departments_df.to_csv('departments.csv', index=False)

print("Data generation for Departments table is complete and saved to departments.csv.")
```

```
Data generation for Departments table is complete and saved to departments.csv.
```

Figure 2: Code to Generate fake data for Departments Table

Positions Table

Five different positions with corresponding job grades were randomly assigned.



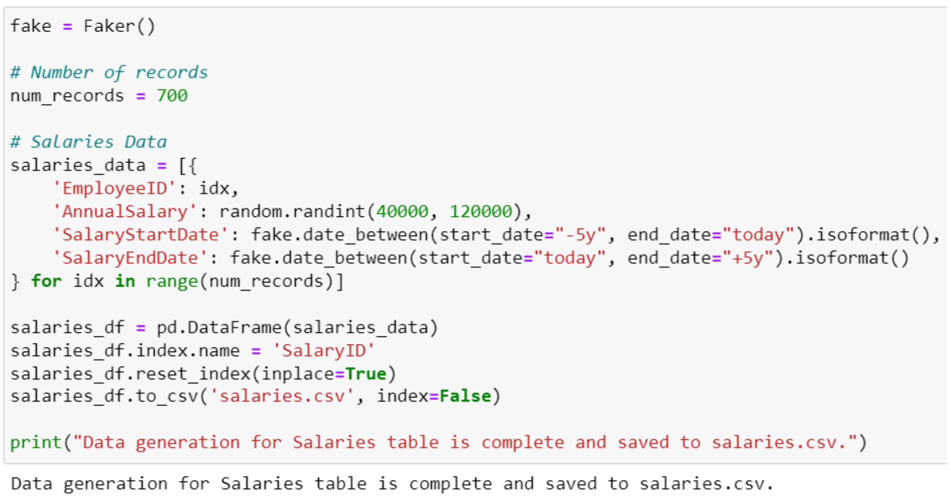
```
# Positions Data
positions_titles = ['Junior', 'Senior', 'Manager', 'Team Lead', 'Director']
job_grades = ['Entry', 'Mid', 'Senior', 'Lead']

positions_df = pd.DataFrame([(title, random.choice(job_grades)) for title in positions_titles],
                            columns=['PositionTitle', 'JobGrade'])

positions_df.index.name = 'PositionID'
positions_df.reset_index(inplace=True)
positions_df.to_csv('positions.csv', index=False)

print("Data generation for Positions table is complete and saved to positions.csv.")
```

```
Data generation for Positions table is complete and saved to positions.csv.
```

Figure 3: Code to Generate fake data for Positions Table

Salaries Table

Randomized annual salary data, along with start and end dates, were generated for each employee.



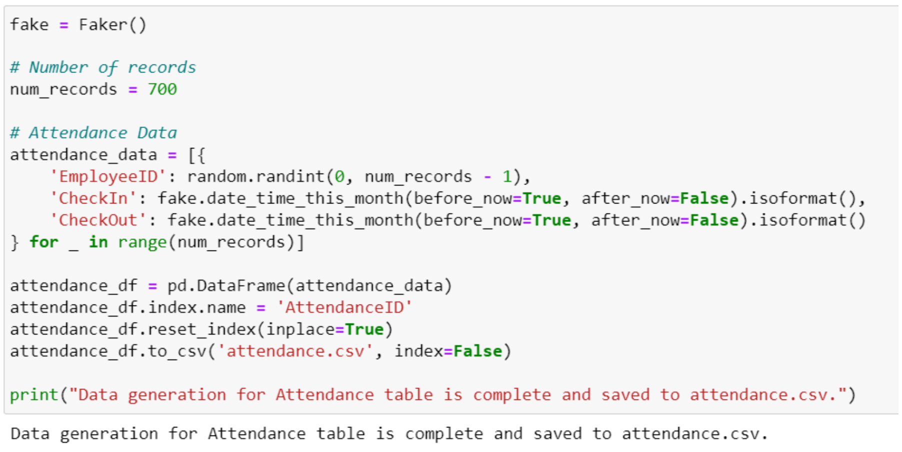
```
fake = Faker()

# Number of records
num_records = 700

# Salaries Data
salaries_data = [{
    'EmployeeID': idx,
    'AnnualSalary': random.randint(40000, 120000),
    'SalaryStartDate': fake.date_between(start_date="-5y", end_date="today").isoformat(),
    'SalaryEndDate': fake.date_between(start_date="today", end_date="+5y").isoformat()
} for idx in range(num_records)]

salaries_df = pd.DataFrame(salaries_data)
salaries_df.index.name = 'SalaryID'
salaries_df.reset_index(inplace=True)
salaries_df.to_csv('salaries.csv', index=False)

print("Data generation for Salaries table is complete and saved to salaries.csv.")
```

```
Data generation for Salaries table is complete and saved to salaries.csv.
```

Figure 4: Code to Generate fake data for Salaries Table

Attendance Table

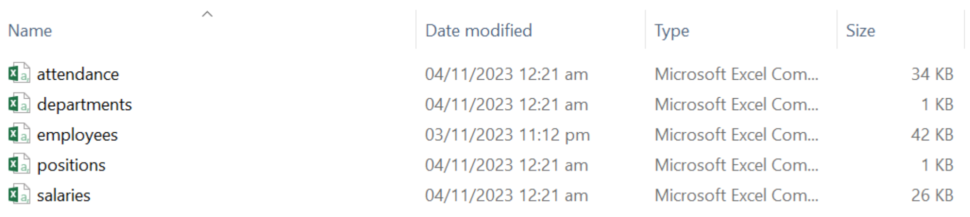
Randomized check-in and check-out data for the past month was generated for employees.


```
fake = Faker()

# Number of records
num_records = 700

# Attendance Data
attendance_data = [{
    'EmployeeID': random.randint(0, num_records - 1),
    'CheckIn': fake.date_time_this_month(before_now=True, after_now=False).isoformat(),
    'CheckOut': fake.date_time_this_month(before_now=True, after_now=False).isoformat()
} for _ in range(num_records)]

attendance_df = pd.DataFrame(attendance_data)
attendance_df.index.name = 'AttendanceID'
attendance_df.reset_index(inplace=True)
attendance_df.to_csv('attendance.csv', index=False)

print("Data generation for Attendance table is complete and saved to attendance.csv.")
```

```
Data generation for Attendance table is complete and saved to attendance.csv.
```

Figure 5: Code to Generate fake data for Attendance Table

The generated data was saved to separate CSV files, ensuring a structured format for data import into our SQL Studio Software. Detail of generated data files is given below.



Name	Date modified	Type	Size
attendance	04/11/2023 12:21 am	Microsoft Excel Com...	34 KB
departments	04/11/2023 12:21 am	Microsoft Excel Com...	1 KB
employees	03/11/2023 11:12 pm	Microsoft Excel Com...	42 KB
positions	04/11/2023 12:21 am	Microsoft Excel Com...	1 KB
salaries	04/11/2023 12:21 am	Microsoft Excel Com...	26 KB

Figure 6: Files of generated data

B. Database Schema

Tables Structure

Employees Table:

- EmployeeID (Primary Key)
- Name (TEXT)
- DepartmentID (Foreign Key)
- PositionID (Foreign Key)
- Birthdate (DATE)
- HireDate (DATE)
- Email (TEXT, Unique)

Departments Table:

- DepartmentID (Primary Key)

- DepartmentName (TEXT)

Positions Table:

- PositionID (Primary Key)
- PositionTitle (TEXT)
- JobGrade (TEXT)

Salaries Table:

- SalaryID (Primary Key)
- EmployeeID (Foreign Key)
- AnnualSalary (INTEGER)
- SalaryStartDate (DATE)
- SalaryEndDate (DATE)

Attendance Table:

- AttendanceID (Primary Key)
- EmployeeID (Foreign Key)
- CheckIn (DATETIME)
- CheckOut (DATETIME)

DFD Diagram

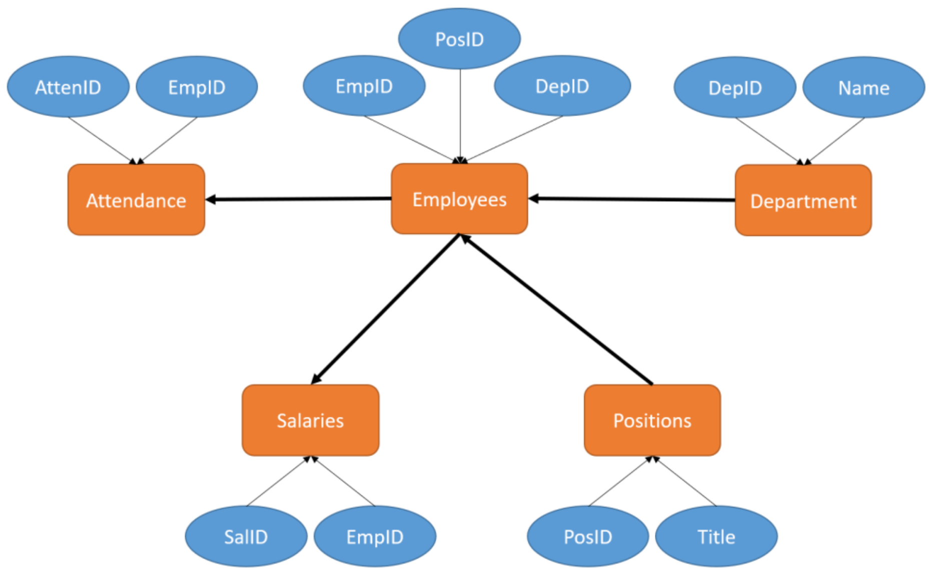


Figure 7: DFD of Database

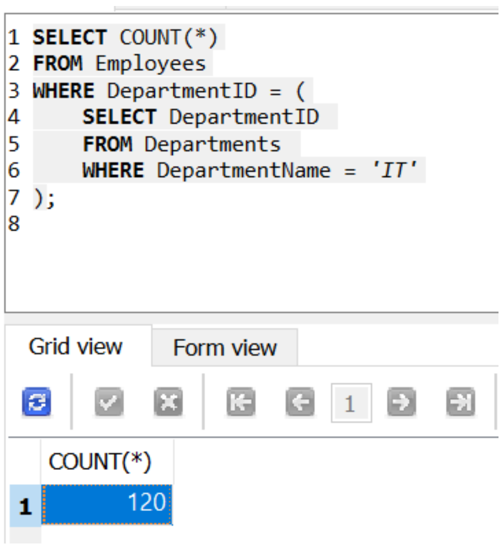
C. Justification for Separate Tables and Ethical Discussion

1. **Separation of Tables**: By normalizing the data and separating them into distinct tables like Employees, Departments, Positions, Salaries, and Attendance, I ensure data integrity, avoid data redundancy, and establish clear relationships between data entities using foreign keys.
2. **Ethical Discussion**: The use of randomized data ensures the privacy and confidentiality of any real individuals. Moreover, as the database grows and real data is added, it's crucial to adhere to data protection regulations, encrypt sensitive data, and have user consent where necessary. Regular audits and backups are also recommended.

D. Example Queries

1. Find the total number of employees in the IT department:

Purpose: This query helps in determining the staffing level of the IT department.



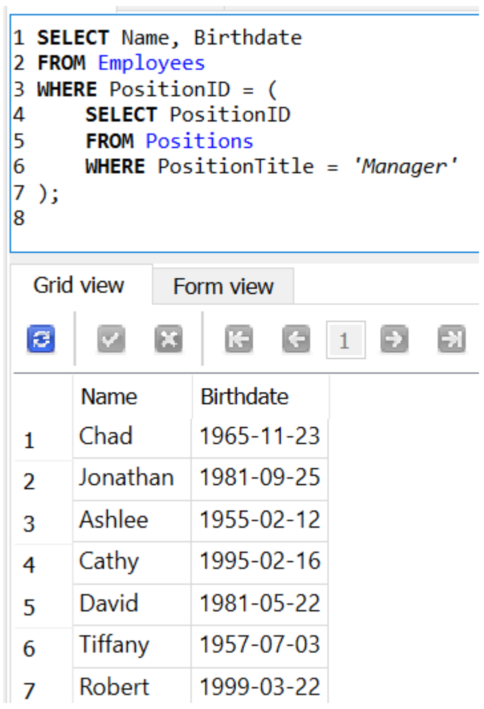
```
SELECT COUNT(*)
FROM Employees
WHERE DepartmentID = (
    SELECT DepartmentID
    FROM Departments
    WHERE DepartmentName = 'IT'
);
```

```
  COUNT(*)
1      120
```

Explanation: The inner query retrieves the DepartmentID of the 'IT' department from the Departments table. The outer query then counts the number of employees who have this DepartmentID in the Employees table.

2. List the names and birthdate of all 'Manager' positions

Purpose: Useful for contacting all managers, e.g., for sending out managerial communications.



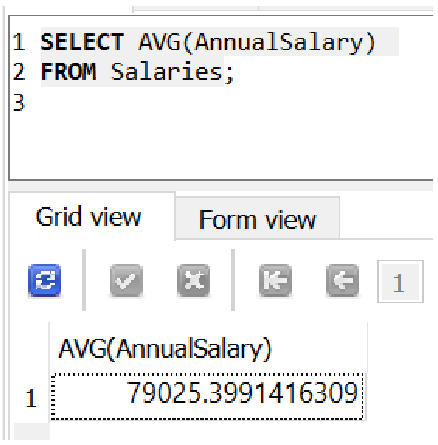
```sql
SELECT Name, Birthdate
FROM Employees
WHERE PositionID = (
    SELECT PositionID
    FROM Positions
    WHERE PositionTitle = 'Manager'
);
```

	Name	Birthdate
1	Chad	1965-11-23
2	Jonathan	1981-09-25
3	Ashlee	1955-02-12
4	Cathy	1995-02-16
5	David	1981-05-22
6	Tiffany	1957-07-03
7	Robert	1999-03-22

Explanation: The inner query fetches the PositionID for the 'Manager' position from the Positions table. The outer query then selects the names and emails of employees who have this PositionID in the Employees table.

3. Determine the average salary of employees

Purpose: To get a quick overview of the average compensation in the company.



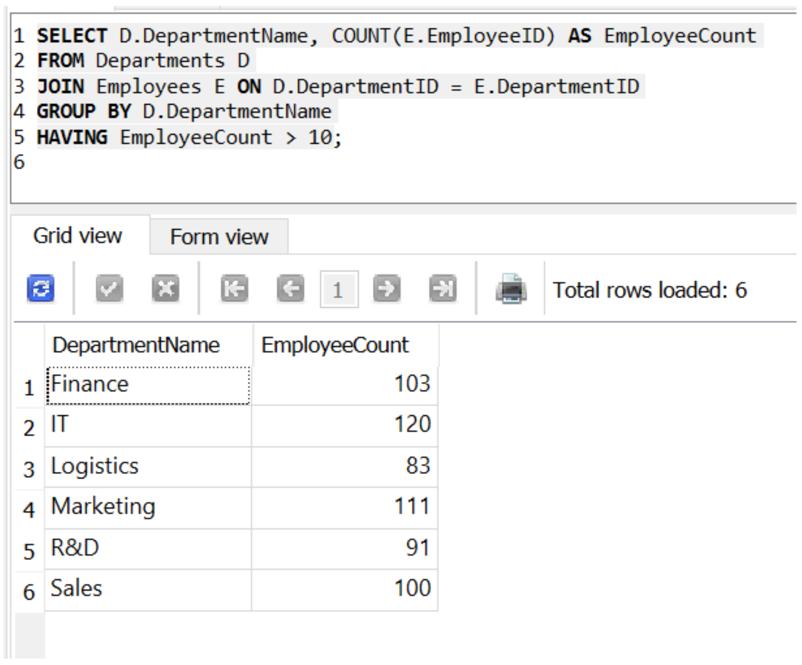
```sql
SELECT AVG(AnnualSalary)
FROM Salaries;
```

	AVG(AnnualSalary)
1	79025.3991416309

Explanation: This query simply takes the average of the AnnualSalary column from the Salaries table, providing an insight into the mean compensation of employees.

4. List the departments with more than 10 employees

Purpose: To identify larger departments that might need more resources or management attention.



```sql
SELECT D.DepartmentName, COUNT(E.EmployeeID) AS EmployeeCount
FROM Departments D
JOIN Employees E ON D.DepartmentID = E.DepartmentID
GROUP BY D.DepartmentName
HAVING EmployeeCount > 10;

```

	DepartmentName	EmployeeCount
1	Finance	103
2	IT	120
3	Logistics	83
4	Marketing	111
5	R&D	91
6	Sales	100

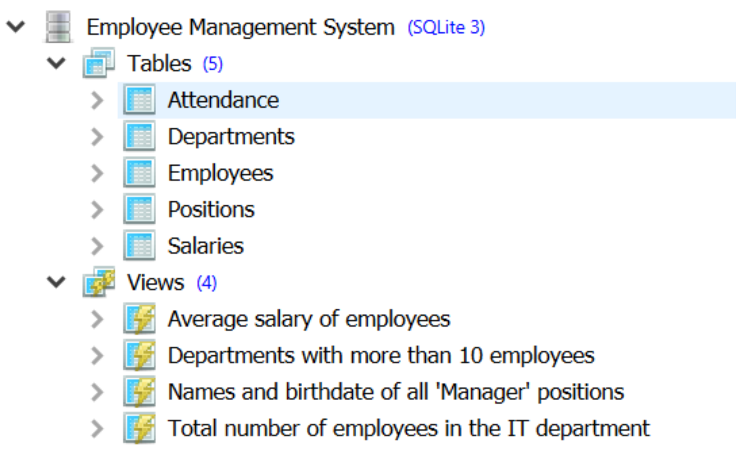
Explanation: This query joins the Departments and Employees tables, groups the results by DepartmentName, and counts the employees in each department. It then filters the results to only include departments with more than 10 employees.

E. Conclusion

The Employee Management System project gave me a chance to learn about the whole process of designing a database, from the initial idea to putting it into action. Using Python, Iwere able to make a big dataset that accurately reflected real-life employee data, including a range of data types and relationships.

The complicated structure of the database, which included many tables and relationships running from primary key constraints to foreign key constraints, showed how deep and useful relational databases can be. These connections not only keep the data safe, but they also make it easier to get complicated data, as the different SQL queries show.

My queries ranged from simple data retrieval to complicated join operations. This showed how powerful SQL can be for getting useful information from structured data. Even though these examples are simple, they show how flexible and powerful relational databases are. They let businesses use their data stores to make smart decisions. Final database was designed in SQLite Studio. This how the database looks like…


Employee Management System (SQLite 3)
Tables (5)
Attendance
Departments
Employees
Positions
Salaries
Views (4)
Average salary of employees
Departments with more than 10 employees
Names and birthdate of all 'Manager' positions
Total number of employees in the IT department

Ethical concerns were also considered and put first when collecting and managing data. Making sure that data is both made up and based on real events shows how to make a database that works without violating people's privacy rights.

Link: https://github.com/awaismuhammad1234/Data-Mining-and-Discovery