SQL Assignment

Abstract:

This report presents the design and implementation of a comprehensive company database tailored to manage employee, department, project, and task information effectively. The database schema encompasses tables for employees, departments, projects, and tasks, establishing pertinent relationships between them.

Introduction:

This report presents the design and implementation details of the company database. The database is designed to manage information about employees, departments, projects, and tasks within a company. The database schema includes tables for employees, departments, projects, and tasks, with appropriate relationships established between them.

Data Generation:

- Data for the databases was generated using Python scripts.
- Employee names were generated using the 'names' library to ensure realistic names.
- Ages were randomly generated within a range of 22 to 65 years to reflect a typical age distribution in the workforce.
- Salaries were randomly generated within a range of $40,000 to $120,000 per year to represent realistic salary distributions.
- Project names were generated by combining the string "Project" with a unique identifier to simulate real project names.
- Task descriptions were generated similarly, with a unique identifier.
- Project and task assignments were made randomly within existing department, project, and employee IDs to ensure realistic relationships.

Code:

```
import sqlite3
import random
import uuid
import names

# Function to generate random salary within a realistic range
def generate_salary():
    return random.randint(40000, 120000)

# Function to generate random age within a realistic range
def generate_age():
    return random.randint(22, 65)

# Function to generate random task priority
def generate_priority():
    priorities = ['Low', 'Medium', 'High']
    return random.choice(priorities)

# Connect to SQLite database
conn = sqlite3.connect('company1.db')
cursor = conn.cursor()

# Create Employees table
cursor.execute('''CREATE TABLE Employees (
                employee_id INTEGER PRIMARY KEY,
                name TEXT NOT NULL,
                age INTEGER,
                department TEXT,
                salary INTEGER
                )''')
```

```python
# Create Departments table
cursor.execute('''CREATE TABLE Departments (
                department_id INTEGER PRIMARY KEY,
                name TEXT NOT NULL
                )''')

# Create Projects table
cursor.execute('''CREATE TABLE Projects (
                project_id INTEGER PRIMARY KEY,
                name TEXT NOT NULL,
                department_id INTEGER,
                FOREIGN KEY (department_id) REFERENCES Departments(department_id
                )''')

# Create Tasks table
cursor.execute('''CREATE TABLE Tasks (
                task_id INTEGER PRIMARY KEY,
                description TEXT NOT NULL,
                priority TEXT,
                project_id INTEGER,
                employee_id INTEGER,
                FOREIGN KEY (project_id) REFERENCES Projects(project_id),
                FOREIGN KEY (employee_id) REFERENCES Employees(employee_id)
                )''')

# Insert departments into Departments table
departments = ['HR', 'Finance', 'IT', 'Marketing', 'Operations']
for dept in departments:
    cursor.execute('''INSERT INTO Departments (name) VALUES (?)''', (dept,))
```

```python
# Generate and insert employee data
for _ in range(1000):
    name = names.get_full_name()
    age = generate_age()
    department = random.choice(departments)
    salary = generate_salary()
    cursor.execute('''INSERT INTO Employees (name, age, department, salary)
                VALUES (?, ?, ?, ?)''', (name, age, department, salary))

# Generate and insert project data
for _ in range(50):
    project_name = "Project " + str(uuid.uuid4())[:8]
    department_id = random.randint(1, len(departments))
    cursor.execute('''INSERT INTO Projects (name, department_id) VALUES (?, ?)''', (project_name, department_id))

# Generate and insert task data
for _ in range(500):
    task_description = "Task " + str(uuid.uuid4())[:8]
    priority = generate_priority()
    project_id = random.randint(1, 50)
    employee_id = random.randint(1, 1000)
    cursor.execute('''INSERT INTO Tasks (description, priority, project_id, employee_id)
                VALUES (?, ?, ?, ?)''', (task_description, priority, project_id, employee_id))

# Commit changes and close connection
conn.commit()
conn.close()
```

Nominal Data:

Nominal data represents categories or labels without any specific order. The ‘name’ column in the employee table and the ‘name’ column in the department table represent nominal data.

Example:

Retrieve the names of employees and their departments.

```
1  SELECT Employees.name AS employee_name, Employees.department, Departments.name AS department_name
2  FROM Employees
3  INNER JOIN Departments ON Employees.department = Departments.name;
4
```

	employee_name	department	department_name
1	Stacy Spain	Marketing	Marketing
2	Susie Fars	HR	HR
3	Joseph Pellot	IT	IT
4	Joan Hansen	HR	HR
5	Jacqueline Marrero	Operations	Operations
6	Amy Lomas	Marketing	Marketing
7	James Kazunas	IT	IT
8	Ralph Marshall	Finance	Finance
9	Vanessa Mccartt	Finance	Finance
10	Leonard Mathews	IT	IT
11	Jaclyn Lundstrom	Finance	Finance
12	Kira Mikels	Operations	Operations
13	Linh Cha	IT	IT
14	Robert Smith	HR	HR
15	David Welby	Finance	Finance
16	John Moody	Operations	Operations
17	Chelsea Crain	IT	IT
18	Susan Henkel	Operations	Operations
19	James Bermudez	Operations	Operations
20	Rosetta Webb	IT	IT
21	Ann Sincell	Marketing	Marketing
22	Jeanne Walker	HR	HR
23	Hazel Tsang	IT	IT

Ordinal data:

Ordinal data represents categories with a specific order or ranking. The ‘priority’ column in the Tasks table represents ordinal data.

Example:

Select tasks with a priority level of ‘High’.

```
SELECT *
FROM Tasks
WHERE priority = 'High';
```

	task_id	description	priority	project_id	employee_id
1	10	Task b5195539	High	6	142
2	12	Task e8c49624	High	16	195
3	14	Task c328b526	High	28	621
4	17	Task 5df3d6dd	High	17	196
5	19	Task 5b490219	High	11	389
6	29	Task 85d80866	High	26	279
7	34	Task 4e5647d4	High	36	576
8	35	Task 5839bba9	High	43	640
9	40	Task ed38d9b8	High	16	569
10	42	Task 273c97d1	High	24	506
11	43	Task e98c2704	High	13	512
12	44	Task 256ed9ab	High	24	449
13	49	Task 60a2e5ac	High	12	573
14	50	Task 075520f5	High	34	123
15	54	Task 4373a6a8	High	2	536
16	67	Task e844a795	High	49	996
17	69	Task 0018d4e5	High	12	909
18	70	Task a67ed43e	High	21	761
19	71	Task 7fd954fa	High	19	496
20	73	Task 7c01921e	High	45	854
21	74	Task 892daa94	High	38	679
22	77	Task 9beb0f1e	High	29	144
23	81	Task 3d23f781	High	11	184

Interval Data:

Interval data represents numerical values where the difference between two values is meaningful, but there is no absolute zero point. The ‘age’ column in the Employees table represents interval data.

Example:

```
SELECT AVG(age) AS average_age
FROM Employees;
```

	average_age
1	43.538

Ratio Data:

Ratio data represents numerical values where both the different values and the ratio of values is meaningful, and there is an absolute zero point. The ‘salary’ column in the employees’ table represents the ratio data.

Example:

Find the highest salary among employees.

```sql
SELECT MAX(salary) AS max_salary
FROM Employees;
```

max_salary
119894

Database Schema:

Name	Type	Schema
Departments		CREATE TABLE Departments (department_id INTEGER PRIMARY KEY, name TEXT NOT NULL)
department_id	INTEGER	"department_id" INTEGER
name	TEXT	"name" TEXT NOT NULL
Employees		CREATE TABLE Employees (employee_id INTEGER PRIMARY KEY, name TEXT NOT NULL, age INTEGER, department TEXT, salary INTEGER)
employee_id	INTEGER	"employee_id" INTEGER
name	TEXT	"name" TEXT NOT NULL
age	INTEGER	"age" INTEGER
department	TEXT	"department" TEXT
salary	INTEGER	"salary" INTEGER
Projects		CREATE TABLE Projects (project_id INTEGER PRIMARY KEY, name TEXT NOT NULL, department_id INTEGER, FOREIGN KEY (department_id) REFERENCES Departments(department_id))
project_id	INTEGER	"project_id" INTEGER
name	TEXT	"name" TEXT NOT NULL
department_id	INTEGER	"department_id" INTEGER
Tasks		CREATE TABLE Tasks (task_id INTEGER PRIMARY KEY, description TEXT NOT NULL, priority TEXT, project_id INTEGER, employee_id INTEGER, FOREIGN KEY (project_id) REFERENCES Projects(project_id), FOREIGN KEY (employee_id) REFERENCES Employees(employee_id))
task_id	INTEGER	"task_id" INTEGER
description	TEXT	"description" TEXT NOT NULL

Name	Type	Schema
priority	TEXT	"priority" TEXT
project_id	INTEGER	"project_id" INTEGER
employee_id	INTEGER	"employee_id" INTEGER

Report Justification:

- The database contains sensitive information about employees, including their salaries and ages. Access to this data should be restricted to authorised personnel only.
- Measures should be taken to ensure data security and privacy, including encryption of sensitive data and access control mechanisms.
- The database should comply with relevant data protection regulations, such as GDPR, to safeguard employee privacy rights.
- The employee table is essential for HR management and payroll processing, as it stores employee details like names, ages, departments, and salaries.
- The department table represents the various departments within the company. Each department has a unique identifier and a name, allowing for easy categorisation and organization of employees and projects.
- The project table tracks information about projects undertaken by the company, including project names and associated departments. This table facilitates project management, resource allocation, and monitoring of project progress.
- The tasks table manages details about individual tasks assigned to employees within projects. Tasks have descriptions, priorities, and associations with specific projects and employees, enabling effective task allocation and tracking.

Example Queries:

Joining Employees with Departments:

This query joins the Employee table with the Departments table to retrieve the names of employees along with their departments.

```
SELECT Employees.name, Employees.department, Departments.name
FROM Employees
INNER JOIN Departments ON Employees.department = Departments.name;
```

	name	department	name
1	Stacy Spain	Marketing	Marketing
2	Susie Fars	HR	HR
3	Joseph Pellot	IT	IT
4	Joan Hansen	HR	HR
5	Jacqueline Marrero	Operations	Operations
6	Amy Lomas	Marketing	Marketing
7	James Kazunas	IT	IT
8	Ralph Marshall	Finance	Finance
9	Vanessa Mccartt	Finance	Finance
10	Leonard Mathews	IT	IT
11	Jaclyn Lundstrom	Finance	Finance
12	Kira Mikels	Operations	Operations
13	Linh Cha	IT	IT
14	Robert Smith	HR	HR
15	David Welby	Finance	Finance
16	John Moody	Operations	Operations
17	Chelsea Crain	IT	IT
18	Susan Henkel	Operations	Operations
19	James Bermudez	Operations	Operations
20	Rosetta Webb	IT	IT
21	Ann Sincell	Marketing	Marketing
22	Jeanne Walker	HR	HR
23	Hazel Tsang	IT	IT

Selecting Employees by Age Range:

This query selects employees within a specified age range.

```sql
SELECT *
FROM Employees
WHERE age BETWEEN 30 AND 40;
```

	employee_id	name	age	department	salary
1	2	Susie Fars	35	HR	113644
2	4	Joan Hansen	37	HR	51968
3	8	Ralph Marshall	31	Finance	116907
4	15	David Welby	37	Finance	76706
5	28	Judith James	35	Finance	58049
6	31	Lawrence Jordan	30	IT	51371
7	37	Judith Huberty	35	Finance	111312
8	61	Maryann Sauer	33	Marketing	109901
9	67	Yvonne Hartley	30	IT	42649
10	73	Robert Schlater	34	Marketing	55241
11	76	Samuel Youngman	30	IT	47411
12	77	Philip Bailey	39	Finance	61380
13	79	Lavonda Williams	31	HR	76630
14	81	Sally Fett	32	Marketing	91504
15	95	Elaine Wilson	30	IT	97331
16	104	Christopher Corp	33	IT	95646
17	111	Steven Tondreau	37	Finance	40905
18	113	Kenneth Taylor	40	HR	60394
19	115	Jessica Smith	35	IT	102125
20	118	Marian Brown	30	Operations	68929
21	135	Norma Schwartz	37	Operations	114960
22	136	Paul Broun	32	IT	100585
23	137	Joni Craig	38	Marketing	107996

Joining Employees with Tasks and Projects:

This query joins the Employees table with the Tasks and Projects tables to retrieve employee names, task descriptions, and project names.

```sql
SELECT Employees.name AS employee_name, Tasks.description AS task_description, Projects.name AS project_name
FROM Employees
INNER JOIN Tasks ON Employees.employee_id = Tasks.employee_id
INNER JOIN Projects ON Tasks.project_id = Projects.project_id;
```

employee_name	task_description	project_name
Rosio Day	Task 105f2054	Project 6bf5bf54
Walter Smith	Task ebad58e5	Project a68ec80d
Dennis Connor	Task d145019e	Project a9f0e627
Roberta Latham	Task 6bd36272	Project 6bf5bf54
Leslie Hicks	Task 19e1b236	Project 243a8e1b
Jessica Koch	Task a43896f6	Project ca4f3f0b
Timothy Dearth	Task cc6009e1	Project 444235e6
Caitlyn Strunk	Task 1f194028	Project 4dc504ad
James Tower	Task 385b3f66	Project 36356be8
Sharron Loske	Task b5195539	Project 0b07ea17
Betty Heroux	Task 53cee13f	Project a86a7731
Kim Hoke	Task e8c49624	Project 1b195544
Michael Holden	Task a1114ca9	Project 2a219505
Joseph Doyle	Task c328b526	Project 987d038b
Pamela Hart	Task 0d17be62	Project d8b9fb4c
Robert Salim	Task fb26a818	Project c695e3fa
Kenneth George	Task 5df3d6dd	Project 0fbb585c
Russell Douglas	Task a861d4bf	Project ee1e29ee
Brandon Boyd	Task 5b490219	Project 46646bb5
Dana Coburn	Task 657949e7	Project 2a219505
Elizabeth Handy	Task cccf66a0	Project 4dc504ad
Kelsey Curry	Task d5c7bae1	Project 2b692eb0
Margaret Pieper	Task 277021bf	Project 2611d980

Calculating Average Salary by Department:

This query calculates the average salary for each department.

```sql
SELECT department, AVG(salary) AS average_salary
FROM Employees
GROUP BY department;
```

	department	average_salary
1	Finance	80931.9223300971
2	HR	81375.5538461539
3	IT	80047.1100917431
4	Marketing	79281.6119402985
5	Operations	77405.9555555556

Conclusion:

In conclusion, the company database serves as a valuable tool for HR management, project management, and decision-making processes within the organisation, enabling efficient data-driven operations while prioritising data security and privacy.

Name: Narendar Vatsavai
Student Id : 22103577