SQL ASSIGNMENT

Name: Bubly Babu

Student ID: 22031115

Topic: Fashion Store Database

I made a database of a fashion store for this assignment. The database for the fashion store is made to seem like how a fashion company may set up its operations. It includes details on orders, clients, fashion designs, and employees. I used numpy and Faker in my Python code to provide data that looked realistic. It resembles a virtual dress-up game for boutiques. First, as seen in the image below, I built a Python dataframe called employee. This dataframe has the employee's id, first and last name, date of birth, department name, and hire date. Whereas the employee ID is nominal data, the birthday is interval data, and the address is ratio data. The entire address could be considered ratio data even though the postcode is nominal in and of itself because it has a significant zero point.

```
[20] !pip install faker
     from faker import Faker
     import numpy as np
     import pandas as pd
     fake = Faker()
```

```
Requirement already satisfied: faker in /usr/local/lib/python3.10/dist-packages (20.0.0)
Requirement already satisfied: python-dateutil>=2.4 in /usr/local/lib/python3.10/dist-packages (from faker) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.4->faker) (1.16.0)
```

```
#Number of samples
n = 1000
# Nominal data: Employee ID
employee_ids = np.random.choice(np.arange(1000, 10000), size=n, replace=False)
# Employee First Name
first_names = np.array([fake.first_name() for i in range(n)])
# Employee Last Name
last_names = np.array([fake.last_name() for i in range(n)])
# Interval data : Birthdate
birth_year = np.random.randint(1970, 1995, n)
birth_month = np.random.randint(1, 13, n)
birth_day = np.random.randint(1, 29, n)
birth_date = [f'{birth_year[i]}-{str(birth_month[i]).zfill(2)}-'
                    f'{str(birth_day[i]).zfill(2)}' for i in range(n)]
# Department
fashion_departments = ['Design', 'Marketing', 'Production', 'Sales', 'Finance']
department = np.random.choice(fashion_departments, size = n, p = (0.25, 0.20, 0.20, 0.20, 0.15))
# Ratio data : Address
characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
postcode = [f'NW{str(i).zfill(2)}'+''.join(np.random.choice(list(characters), size=2)) for i in range(1, 11)]
postcode_data = np.random.choice(postcode, size = n)
```

```
#Hire Date
hire_year = np.random.randint(2013, 2023, n)
hire_month = np.random.randint(1, 13, n)
hire_day = np.random.randint(1, 29, n)
hire_date = [f'{hire_year[i]}-{str(hire_month[i]).zfill(2)}-'
                    f'{str(hire_day[i]).zfill(2)}' for i in range(n)]

df_employees = pd.DataFrame({
    'Employee ID' : employee_ids,
    'First Name' : first_names,
    'Last Name' : last_names,
    'Birth Date' : birth_date,
    'Department Name' : department,
    'Address' : postcode_data,
    'Hire Date' : hire_date
})
df_employees
```

The output of this dataframe is as below.

	Employee ID	First Name	Last Name	Birth Date	Department Name	Address	Hire Date
0	5421	Wendy	Livingston	1983-05-09	Finance	NW09QN	2021-10-05
1	7309	Chad	Olsen	1980-07-28	Production	NW03XI	2022-03-18
2	3961	Aimee	Hale	1992-04-11	Finance	NW09QN	2015-08-16
3	1069	Cheryl	Watkins	1984-03-09	Design	NW01VV	2014-12-23
4	2131	Margaret	Lee	1994-04-19	Sales	NW07EU	2014-07-20
...	...	...	...	...	...	...	...
995	5948	Marisa	Carrillo	1987-04-24	Finance	NW09QN	2018-11-20
996	1497	Jonathan	Ray	1972-04-27	Production	NW02OX	2014-09-09
997	8472	Ryan	Graham	1975-05-26	Sales	NW05JX	2015-03-11
998	6873	Henry	Black	1986-05-17	Sales	NW04EC	2020-02-14
999	9551	Christopher	Dougherty	1988-06-15	Production	NW10PW	2021-06-11

1000 rows × 7 columns

The information regarding fashion designs and their details is provided in the second Python Dataframe below. It includes information on the design name, design ID, and employee ID of those who created the designs.

```python
# Design id
design_ids = np.random.choice([str(i).zfill(4) for i in range(1, 1001)], size = n, replace = False)
# Design category
design_names = np.random.choice(['Spring Collection', 'Summer Line', 'Fall Fashion', 'Holiday Glam',\
                                 'Casual Chic', 'Elegant Evening', 'Street Style', 'Vintage Vibes',\
                                 'Urban Glam','Cozy Winter'], n)
# Designer
design_department_employees = df_employees[df_employees['Department Name'] == "Design"]
designer = np.random.choice(design_department_employees['Employee ID'], n)

df_design = pd.DataFrame({
    'Design ID' : design_ids,
    'Design Names' : design_names,
    'Employee ID' : designer
})
df_design
```

	Design ID	Design Names	Employee ID
0	0444	Elegant Evening	8005
1	0944	Spring Collection	2326
2	0446	Holiday Glam	4193
3	0115	Urban Glam	8019
4	0863	Elegant Evening	2090
...	...	...	...
995	0619	Vintage Vibes	5621
996	0237	Holiday Glam	7956
997	0600	Spring Collection	8550
998	0453	Vintage Vibes	6847
999	0507	Spring Collection	6415

1000 rows × 3 columns

I create the third dataframe and then add the client's information to it. Along with the client's membership type, it includes the client's name, client ID, phone number, and email address. Seventy of them had nan values supplied in the client email data. The membership type in this fashion store database is an ordinal data type. The output of the dataframe and the Python code are shown below.

```
# Client id
id = [str(i).zfill(5) for i in range(10000, 99999)]
client_id = np.random.choice(id, size = n, replace = False)
# Client names
client_names = [fake.name() for i in range(n)]
# Contact Email
domains = ['mac.com', 'gmail.com', 'hotmail.com', 'yahoo.com', 'outlook.com.au']
client_emails = [f"{name.split()[0].lower()}.{name.split()[1].lower()}\
                 @{np.random.choice(domains)}" for name in client_names]
# Randomly select 50 indices to set to NaN
n_points = 70
random_indices = np.random.choice(n, n_points, replace=False)
client_emails = np.array(client_emails)
client_emails[random_indices] = np.nan
# Contact Number
country_code = '+44'
```

```
client_contact = np.array([f"{country_code} {fake.random_number(10)}" for i in range(n)])
# Ordinal data : Membership
membership_types = np.random.choice(['Basic', 'Premium', 'VIP'], n, p=(0.4, 0.3, 0.3))

df_clients = pd.DataFrame({
    'Client ID' : client_id,
    'Client Names' : client_names,
    'Contact Email' : client_emails,
    'Contact Number' : client_contact,
    'Membership' : membership_types
})
df_clients
```

	Client ID	Client Names	Contact Email	Contact Number	Membership
0	60360	Caleb Johns	caleb.johns @hotmail.com	+44 146569779	Premium
1	68301	Holly Williams	holly.williams @mac.com	+44 5404348340	VIP
2	19202	Karen Perry	nan	+44 8479378655	VIP
3	69143	Mark Smith	mark.smith @gmail.com	+44 9325823801	Premium
4	39511	Stephanie Hernandez	stephanie.hernandez @hotmail.com	+44 8618018685	Basic
...	...	...	...	...	...
995	66685	Joseph Gonzalez	joseph.gonzalez @hotmail.com	+44 643378502	Basic
996	90460	Troy Fisher	troy.fisher @hotmail.com	+44 4771837191	Basic
997	91360	Joshua Moore	joshua.moore @mac.com	+44 3775456434	Premium
998	80132	Wanda Hale	wanda.hale @mac.com	+44 3752172957	VIP
999	35633	Ashley Rogers	ashley.rogers @outlook.com.au	+44 8752985558	VIP

1000 rows × 5 columns

The last dataframe I created for this project is the order dataframe. Among the information it carries are order ID, design ID, customer ID, order date, client rating, and compound key. The compound key is created using the order ID and design ID, giving each order placed by the client a unique identity. The steps taken to create the dataframe and its outcomes are described below.

```
# Order id
order_ids = np.random.choice([str(i).zfill(3) for i in range(1, n + 1)],\
                             size = n, replace = False)
# Designs ordered
designs_ordered = np.random.choice(design_ids, n)
# Clients ordered
clients_ordered = np.random.choice(client_id, n)
# Order date
order_dates = np.random.choice(pd.date_range(start='2022-01-01',\
                                             end='2023-01-01', freq='D'), n)
# Rating
rating = np.random.uniform(1,10,n).astype(int)

df_order = pd.DataFrame({
    'Order ID' : order_ids,
    'Desgin ID' : designs_ordered,
    'Client ID' : client_id,
    'Order Date' : order_dates,
    'Client Rating' : rating
})
# Add a compound key combining OrderID and DesignID
df_order['CompoundKey'] = df_order['Order ID'].astype(str) + '_'\
         + df_order['Desgin ID'].astype(str)
df_order
```

	Order ID	Desgin ID	Client ID	Order Date	Client Rating	CompoundKey
0	336	0755	60360	2022-05-09	2	336_0755
1	155	0462	68301	2022-11-17	2	155_0462
2	441	0200	19202	2022-12-08	4	441_0200
3	080	0344	69143	2022-09-25	2	080_0344
4	771	0962	39511	2022-06-08	1	771_0962
...	...	...	...	...	...	...
995	864	0766	66685	2022-02-16	9	864_0766
996	351	0076	90460	2022-05-19	5	351_0076
997	369	0062	91360	2022-07-05	3	369_0062
998	769	0451	80132	2022-05-06	1	769_0451
999	974	0811	35633	2022-06-20	8	974_0811

1000 rows × 6 columns

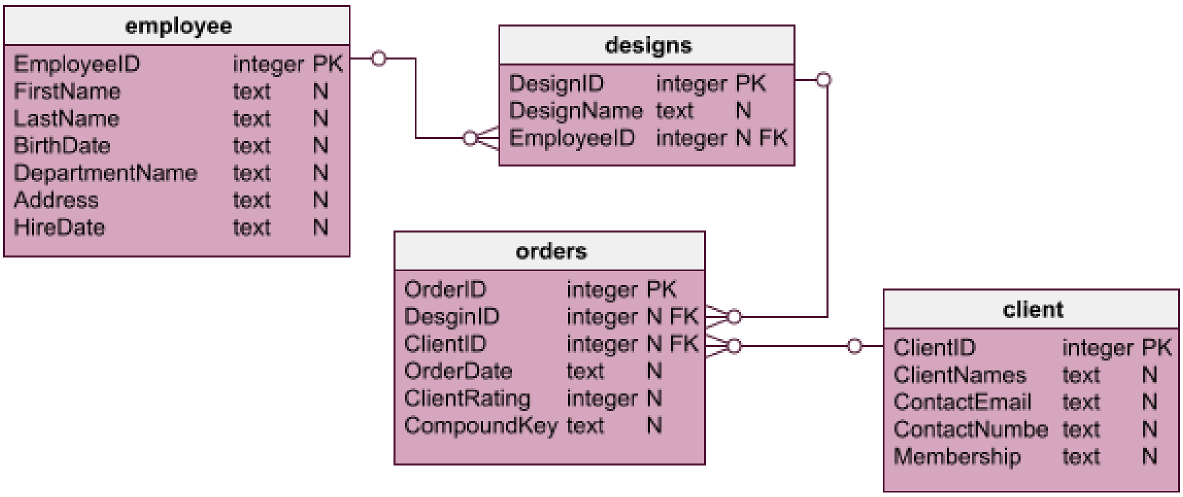
Finally, I save each dataframe as a CSV file after assigning indexes to them. Here is the code to use for this.

```
[26] # Example index
     df_employees.set_index('Employee ID', inplace=True)
     df_design.set_index('Design ID', inplace=True)
     df_clients.set_index('Client ID', inplace=True)
     df_order.set_index('Order ID', inplace=True)
```

```
[27] # Saving Dataframes to CSV
     df_employees.to_csv('employee.csv')
     df_design.to_csv('designs.csv')
     df_clients.to_csv('client.csv')
     df_order.to_csv('orders.csv')
```

My next move in this assignment is to import these CSV files into SQLite DB Browser in order to establish a database. Next, the table was altered by setting the foreign keys, unique values, and primary keys—not null values.

A schema, as used in database architecture, is a logical diagram that outlines the table definitions, constraints, and relationships as well as the organisation and structure of data within a database. Table associations are constructed by the use of foreign key references, in which the primary key of one table is utilised as a foreign key in another. For instance, the "Employee ID" in the "design" field functions as a foreign key that links designs to particular employees by referencing the "Employee ID" in the "employees" table. The "Client ID" and "Design ID" in the "orders" record similarly build associations between designs, clients, and orders by referencing the appropriate primary keys in the "design" and "clients" columns. The schema of the database that I created is given below.



Here are a few sample queries and their results that I ran through the SQLite DB Browser's Execute SQL tab.

1. This query is designed to retrieve information about the details of the first 10 employees from the "employee" table who belong to the 'Marketing' department.

SQL 1 SQL 2 SQL 3 SQL 4 SQL 5

```
SELECT *
FROM employee
WHERE DepartmentName = 'Marketing'
LIMIT 10;
```

	EmployeeID	FirstName	LastName	BirthDate	DepartmentName	Address	HireDate
1	1038	John	Powell	1976-08-05	Marketing	NW07FO	2019-10-15
2	1065	Sabrina	Peterson	1984-08-02	Marketing	NW04NH	2015-06-22
3	1072	Sandy	Myers	1978-11-04	Marketing	NW01CT	2016-09-09
4	1121	Christopher	Turner	1978-10-19	Marketing	NW05KF	2016-12-25
5	1136	Cristian	Walton	1985-07-07	Marketing	NW02OK	2016-03-09
6	1182	Lauren	Castillo	1989-05-20	Marketing	NW06ZM	2018-06-04
7	1190	Linda	Christensen	1980-03-20	Marketing	NW03LS	2017-04-18
8	1305	Pamela	Clements	1974-03-04	Marketing	NW06ZM	2021-06-02
9	1351	David	Walker	1982-05-22	Marketing	NW06ZM	2022-12-15
10	1358	Tina	Moreno	1981-12-28	Marketing	NW01CT	2021-08-19

2. This SQL query retrieves the client names and their respective order counts from the "orders" and "client" tables, joining them based on the 'ClientID,' and presents the top 10 clients with the highest order counts, ordered in descending order.

SQL 1 SQL 2 SQL 3 SQL 4 SQL 5

```sql
SELECT ClientNames, COUNT(OrderID) AS OrderCount
FROM orders
JOIN client
ON client.ClientID = orders.ClientID
GROUP BY ClientNames
ORDER BY OrderCount DESC
LIMIT 10;
```

	ClientNames	OrderCount
1	Richard Miller	2
2	Rachel Robinson	2
3	Kevin Jones	2
4	Jeremiah Gonzalez	2
5	Jeffrey Collins	2
6	Zachary Carpenter	1
7	Yvonne Burgess	1
8	Yvette Cook	1
9	Wyatt Parsons	1
10	William Rodriguez	1

3. This SQL query selects employees, including their first and last names, and the corresponding design names, were involved in design projects, and were hired between January 1, 2020, and December 31, 2022, limited to the first 15 records.

SQL 1 SQL 2 SQL 3 SQL 4 SQL 5

```sql
SELECT FirstName, LastName, DesignNames, HireDate
FROM employee
JOIN designs
ON employee.EmployeeID = designs.EmployeeID
WHERE HireDate BETWEEN '2020-01-01' AND '2022-12-31'
LIMIT 15;
```

	FirstName	LastName	DesignNames	HireDate
1	Tonya	Waller	Cozy Winter	2021-10-22
2	Ernest	Brown	Cozy Winter	2021-11-01
3	Shawn	Warner	Elegant Evening	2021-11-10
4	Travis	Morgan	Vintage Vibes	2021-07-23
5	Wayne	Johnson	Urban Glam	2020-09-18
6	Vanessa	Perez	Casual Chic	2022-08-04
7	Alexandra	Davis	Urban Glam	2022-02-07
8	Jerome	Evans	Street Style	2021-09-15
9	Brenda	Andrews	Urban Glam	2021-12-13
10	Wayne	Johnson	Vintage Vibes	2020-09-18
11	Ryan	Foster	Holiday Glam	2022-05-15
12	Patricia	Martinez	Fall Fashion	2022-10-26
13	Debbie	Smith	Spring Collection	2020-02-18
14	Wendy	Zamora	Spring Collection	2022-09-03
15	Lisa	Jones	Holiday Glam	2021-10-19

4. This SQL query selects all columns from the "orders" table corresponding to the compound key '009_0286' formed by concatenating 'Order ID' and 'Design ID'.

SQL 1 | SQL 2 | SQL 3 | SQL 4 | SQL 5

```
SELECT *
FROM client
WHERE ClientNames LIKE '%JAMES%' AND Membership = 'VIP';
```

	ClientID	ClientNames	ContactEmail	ContactNumber	Membership
1	12623	James Sawyer	james.sawyer@hotmail.com	+44 1267197903	VIP
2	14118	James Price	james.price@hotmail.com	+44 8243143991	VIP
3	61401	James Todd	james.todd@hotmail.com	+44 3201704156	VIP
4	66557	James Garcia	james.garcia@yahoo.com	+44 6720326692	VIP
5	82697	Kaitlyn James	kaitlyn.james@gmail.com	+44 4689039062	VIP
6	92180	Caitlyn James	caitlyn.james@gmail.com	+44 7661435509	VIP

5. This SQL query retrieves the details of clients, with names containing 'JAMES' (case-insensitive) and holding a 'VIP' membership.

SQL 1 | SQL 2 | SQL 3 | SQL 4 | SQL 5

```
SELECT *
FROM client
WHERE ClientNames LIKE '%JAMES%' AND Membership = 'VIP';
```

	ClientID	ClientNames	ContactEmail	ContactNumber	Membership
1	12623	James Sawyer	james.sawyer@hotmail.com	+44 1267197903	VIP
2	14118	James Price	james.price@hotmail.com	+44 8243143991	VIP
3	61401	James Todd	james.todd@hotmail.com	+44 3201704156	VIP
4	66557	James Garcia	james.garcia@yahoo.com	+44 6720326692	VIP
5	82697	Kaitlyn James	kaitlyn.james@gmail.com	+44 4689039062	VIP
6	92180	Caitlyn James	caitlyn.james@gmail.com	+44 7661435509	VIP