Advanced Databases UFCFU3-15-3: Task 1

Arthur Milner (21035478)

Normalisation

Before converting the given data file into an ERD, and consequently an SQL database, it is important to first ensure the data is in 3rd Normal Form to remove data redundancy and minimise data dependency **(Datacamp, 2024)**.

1st Normal Form

To get the data into 1st normal form the data must be atomic, have no repeating groups, no duplicate rows, each column must have a unique name, and each row must have a unique ID. In the case study data, the columns all have a unique name, and there aren't any duplicate rows.

It is arguable, however, that whilst the example data does show name/email to be a unique identifier it would be beneficial to add a Person ID, this helps managing foreign keys as a name/email might change, but the ID will always remain the same. I will also split the name column into First Name and Last name for both persons and neighbours to promote atomicity, although not essential.

There is the assumption a person can have multiple addresses, and multiple of the same type of favourite, so a separate table will be created for addresses and favourites, to prevent repeating groups. The setup of the favourite table prevents null values in a situation such as when a person has a favourite drink but not a favourite book/activity. The neighbours will be in the address table, as there are always 2 neighbours attached to an address, so it does not create a repeating group in this format. The person ID and address/favourite ID will combine to create a composite key here.

Below is the table setup for 1st normal form with 5 sample rows.

Person Table

Person ID (PK)	First Name	Last Name	Email	DoB
1	Person	1	Person1@email.com	15/03/1995
2	Person	2	Person2@email.com	22/06/1993
3	Person	3	Person3@email.com	10/09/1991
4	Person	4	Person4@email.com	05/12/1998
5	Person	5	Person5@email.com	30/11/1983

Address Table

Person ID (FK)	Address ID (PK)	Street	City	Country	Zip Code	Neighbour 1 First Name	Neighbour 1 Last Name	Neighbour 1 Email	Neighbour 2 First Name	Neighbour 2 Last Name	Neighbour 2 Email

1	1	12 Maple St	London	England	E1 6AN	Neighbor	A	neighborA@email.com	Neighbor	B	neighborB@email.com
2	2	45 Oak Ave	Manchester	England	M1 2WD	Neighbor	C	neighborC@email.com	Neighbor	D	neighborD@email.com
3	3	89 Pine Rd	Birmingham	England	B1 1AB	Neighbor	E	neighborE@email.com	Neighbor	F	neighborF@email.com
4	4	23 Birch St	Edinburgh	Scotland	EH1 1YZ	Neighbor	G	neighborG@email.com	Neighbor	H	neighborH@email.com
5	5	67 Cedar Ln	Bristol	England	BS1 3XE	Neighbor	I	neighborI@email.com	Neighbor	J	neighborJ@email.com

Favourite Table

Person ID (FK)	Favourite ID (PK)	Favourite Type	Favourite Name
1	1	Book	A New Beginning
1	2	Drink	Lemonade
1	3	Activity	Outdoor Hiking
2	4	Book	The Road to Success
2	5	Drink	Coffee

2nd Normal Form

Now the data is in 1st normal form I can convert it into 2nd normal form. This involves removing partial dependencies.

There are partial dependencies present between Person ID and other respective IDs in the Address (Address ID) and Favourite (Favourite ID) tables. For example, in the Address table both address and neighbour details are dependent only on Address ID, not Person ID. In the Favourite table, favourite details are dependent only on Favourite ID, not Person ID. To fix this I will be making junction tables in which one row will represent the relationship between a person and an address/favourite, with the IDs combining to form a composite key. This also minimises data redundancy, particularly when dealing with many records. For example, where persons might share the same address, you would not need to store the address details twice and only create a row in the junction table.

Below are the updated tables in 2nd normal form with 5 sample rows.

Person Table

Person ID (PK)	First Name	Last Name	Email	DoB
1	Person	1	Person1@email.com	15/03/1995
2	Person	2	Person2@email.com	22/06/1993
3	Person	3	Person3@email.com	10/09/1991
4	Person	4	Person4@email.com	05/12/1998

5	Person	5	Person5@email.com	30/11/1983

Address Table

Address ID (PK)	Street	City	Country	Zip Code	Neighbour 1 First Name	Neighbour 1 Last Name	Neighbour 1 Email	Neighbour 2 First Name	Neighbour 2 Last Name	Neighbour 2 Email
1	12 Maple St	London	England	E1 6AN	Neighbor	A	neighborA@email.com	Neighbor	B	neighborB@email.com
2	45 Oak Ave	Manchester	England	M1 2WD	Neighbor	C	neighborC@email.com	Neighbor	D	neighborD@email.com
3	89 Pine Rd	Birmingham	England	B1 1AB	Neighbor	E	neighborE@email.com	Neighbor	F	neighborF@email.com
4	23 Birch St	Edinburgh	Scotland	EH1 1YZ	Neighbor	G	neighborG@email.com	Neighbor	H	neighborH@email.com
5	67 Cedar Ln	Bristol	England	BS1 3XE	Neighbor	I	neighborI@email.com	Neighbor	J	neighborJ@email.com

Person Address Table

Person ID (FK)	Address ID (FK)
1	1
2	2
3	3
4	4
5	5

Favourite Table

Favourite ID (PK)	Favourite Type	Favourite Name
1	Book	A New Beginning
2	Drink	Lemonade
3	Activity	Outdoor Hiking
4	Book	The Road to Success
5	Drink	Coffee

Person Favourite Table

Person ID (FK)	Favourite ID (FK)
1	1
1	2
1	3
2	4
2	5

3rd Normal Form

Moving towards 3rd normal form I am looking to remove transitive dependencies; I cannot identify any signs of transitive dependencies in most of the tables. Addresses, however, contains transitive dependencies.

Address ID determines Zip Code, Zip Code can determine City, and Address ID determines City. Because of this I have created a Zip Code, City, and a Country table. Whilst this increases query complexity by requiring more joins, it greatly reduces data redundancy as addresses often share the same details, particularly city/country, but more importantly it removes the transitive dependency and follows 3rd normal form. Splitting address makes the current design more scalable when considering large datasets for this very same reason.

Finally, there is another transitive dependency seen in the Address table, Address ID -> Neighbour Email, Neighbour Email -> Neighbour Name, and Address ID -> Neighbour Name, causing a transitive dependency. It also makes sense to split neighbour into a separate table as a neighbour could theoretically belong to multiple addresses.

Below are the updated tables in 3rd normal form with the full dataset.

Person Table

Person ID (PK)	First Name	Last Name	Email	DoB
1	Person	1	Person1@email.com	15/03/1995
2	Person	2	Person2@email.com	22/06/1993
3	Person	3	Person3@email.com	10/09/1991
4	Person	4	Person4@email.com	05/12/1998
5	Person	5	Person5@email.com	30/11/1983
6	Person	6	Person6@email.com	18/07/1989
7	Person	7	Person7@email.com	25/04/1996
8	Person	8	Person8@email.com	09/01/1990
9	Person	9	Person9@email.com	17/08/1993
10	Person	10	Person10@email.com	22/10/1997
11	Person	11	Person11@email.com	13/05/1992
12	Person	12	Person12@email.com	27/02/1986
13	Person	13	Person13@email.com	25/11/1991
14	Person	14	Person14@email.com	01/02/1987
15	Person	15	Person15@email.com	12/08/1984
16	Person	16	Person16@email.com	09/03/1990
17	Person	17	Person17@email.com	17/11/1995
18	Person	18	Person18@email.com	20/06/1994
19	Person	19	Person19@email.com	11/12/1992
20	Person	20	Person20@email.com	25/09/1988

Address Table

Address ID (PK)	Street	Zip Code ID (FK)	Neighbour 1 ID (FK)	Neighbour 2 ID (FK)
1	12 Maple St	1	1	2
2	45 Oak Ave	2	3	4
3	89 Pine Ave	4	5	6
4	23 Birch St	5	7	8
5	67 Cedar Ln	6	9	10
6	56 Elm St	7	11	12
7	12 Maple St	8	13	14
8	89 Oak Dr	9	15	16
9	123 Pine Rd	10	17	18
10	15 Elm St	11	19	20
11	78 Oak Ln	12	21	22
12	56 Birch Rd	13	23	24
13	10 Holy St	15	25	26
14	34 Willow Rd	14	27	28
15	78 Cedar Ave	16	29	30
16	45 Maple Rd	17	31	32
17	23 Birch Ave	18	33	34
18	12 Elm Blvd	19	35	36
19	56 Oak Rd	20	37	28
20	90 Pine Ave	3	39	40

Neighbour Table

Neighbour ID (PK)	Neighbour First Name	Neighbour Last Name	Neighbour Email
1	Neighbor	A	neighborA@email.com
2	Neighbor	B	neighborB@email.com
3	Neighbor	C	neighborC@email.com
4	Neighbor	D	neighborD@email.com
5	Neighbor	E	neighborE@email.com
6	Neighbor	F	neighborF@email.com
7	Neighbor	G	neighborG@email.com
8	Neighbor	H	neighborH@email.com
9	Neighbor	I	neighborI@email.com
10	Neighbor	J	neighborJ@email.com
11	Neighbor	K	neighborK@email.com
12	Neighbor	L	neighborL@email.com
13	Neighbor	M	neighborM@email.com
14	Neighbor	N	neighborN@email.com
15	Neighbor	O	neighborO@email.com
16	Neighbor	P	neighborP@email.com
17	Neighbor	Q	neighborQ@email.com
18	Neighbor	R	neighborR@email.com
19	Neighbor	S	neighborS@email.com

20	**Neighbor**	**T**	**neighborT@email.com**
21	**Neighbor**	**U**	**neighborU@email.com**
22	**Neighbor**	**V**	**neighborV@email.com**
23	**Neighbor**	**W**	**neighborW@email.com**
24	**Neighbor**	**X**	**neighborX@email.com**
25	**Neighbor**	**Y**	**neighborY@email.com**
26	**Neighbor**	**Z**	**neighborZ@email.com**
27	**Neighbor**	**AA**	**neighborAA@email.com**
28	**Neighbor**	**AB**	**neighborAB@email.com**
29	**Neighbor**	**AC**	**neighborAC@email.com**
30	**Neighbor**	**AD**	**neighborAD@email.com**
31	**Neighbor**	**AE**	**neighborAE@email.com**
32	**Neighbor**	**AF**	**neighborAF@email.com**
33	**Neighbor**	**AG**	**neighborAG@email.com**
34	**Neighbor**	**AH**	**neighborAH@email.com**
35	**Neighbor**	**AI**	**neighborAI@email.com**
36	**Neighbor**	**AJ**	**neighborAJ@email.com**
37	**Neighbor**	**AK**	**neighborAK@email.com**
38	**Neighbor**	**AL**	**neighborAL@email.com**
39	**Neighbor**	**AM**	**neighborAM@email.com**
40	**Neighbor**	**AN**	**neighborAN@email.com**

Zip Code Table

Zip Code ID (PK)	Zip Code	City ID (FK)
1	E1 6AN	**1**
2	M1 2WD	**2**
3	B1 1AB	**10**
4	EH1 1YZ	**3**
5	BS1 3XE	**4**
6	L1 1AA	**5**
7	G1 2TF	**6**
8	LS1 3AB	**7**
9	NE1 2AB	**8**
10	CF10 3AF	**9**
11	S1 4GT	**10**
12	NG1 2PB	**11**
13	CF10 2NF	**12**
14	EH1 1AB	**4**
15	CB1 2SE	**10**
16	OX2 6TP	**13**
17	SO14 3HL	**14**
18	LE1 3PL	**15**
19	NR1 4BE	**16**
20	CF10 3BC	**17**

City Table

City ID (PK)	City	Country ID (FK)

1	London	1
2	Manchester	1
3	Birmingham	1
4	Edinburgh	2
5	Bristol	1
6	Liverpool	1
7	Glasgow	2
8	Leeds	1
9	Newcastle	1
10	Cardiff	3
11	Sheffield	1
12	Nottingham	1
13	Cambridge	1
14	Oxford	1
15	Southampton	1
16	Leicester	1
17	Norwich	1

Country Table

Country ID (PK)	Country Name
1	England
2	Scotland
3	Wales

Person Address Table

Person ID (FK)	Address ID (FK)
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20

Favourite Table

Favourite ID (PK)	Favourite Type	Favourite Name
1	Book	A New Beginning
2	Book	The Road to Success
3	Book	Endless Possibilities
4	Book	Journey of Life
5	Book	The Adventure Continues
6	Book	Finding Inner Peace
7	Book	Exploring New Horizons
8	Book	The Great Journey
9	Book	The Power of Change
10	Book	New Beginnings Await
11	Book	Wandering Souls
12	Book	Freedom and Choice
13	Book	Chasing Dreams
14	Book	The Endless Journey
15	Book	The Future Ahead
16	Book	The Path to Glory
17	Book	Life’s Adventure
18	Book	Into the Wild
19	Drink	Lemonade
20	Drink	Coffee
21	Drink	Smoothie
22	Drink	Iced Tea
23	Drink	Green Tea
24	Drink	Coconut Water
25	Drink	Fruit Juice
26	Drink	Water
27	Drink	Hot Chocolate
28	Drink	Fruit Smoothie
29	Drink	Sparkling Water
30	Drink	Herbal Tea
31	Drink	Iced Coffee
32	Activity	Outdoor Running
33	Activity	Hiking
34	Activity	Swimming
35	Activity	Traveling
36	Activity	Gardening
37	Activity	Reading
38	Activity	Cycling
39	Activity	Skiing
40	Activity	Jogging
41	Activity	Rock Climbing
42	Activity	Yoga
43	Activity	Running

Person Favourite Table

Person ID (FK)	Favourite ID (FK)
1	1
2	2
3	3
4	4
5	5
20	5
6	6
7	7
8	8
9	9
10	10
13	10
11	11
12	12
14	13
15	14
16	15
17	16
18	17
19	18
1	19
14	19
2	20
3	21
13	21
4	22
5	23
17	23
6	24
18	24
7	25
16	25
8	26
20	26
9	27
10	28
11	29
12	30
19	30
15	31
1	32

2	**33**
8	**33**
13	**33**
18	**33**
20	**33**
3	**34**
19	**34**
4	**35**
5	**36**
17	**36**
6	**37**
7	**38**
15	**38**
9	**39**
10	**40**
11	**41**
12	**42**
16	**42**
14	**43**

ER Diagram

To assist in my design, I have created an ER diagram **(Fig. 1)** to visually display the tables, fields, relations, and constraints to be reflected within the database, such as the ON UPDATE and ON DELETE parameters to maintain referential integrity. I have not made city name unique as a city can be under the same name but in a different country, this is the same for zip code as a zip code can cover multiple cities.

Junction tables have ON DELETE CASCADE on the person_id so relationships are not kept if the person is removed. ON DELETE RESTRICT on the address_id and favourite_id prevents addresses and favourites from being deleted if they are in use by a person, this is also applied to the references inside the address table to prevent orphaned references. ON UPDATE CASCADE ensures key updates are reflected.

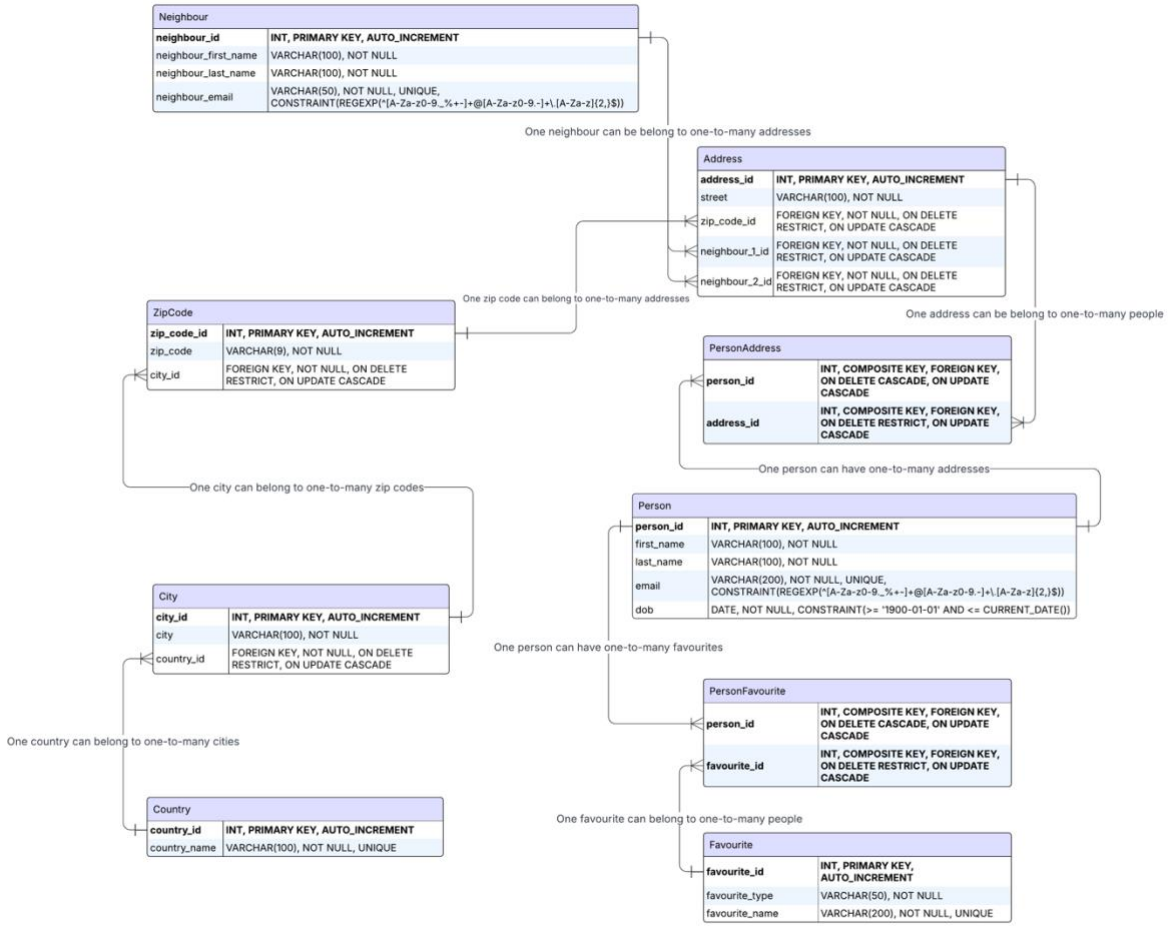


Figure 1 ERD for the database

Implementation

Creating the Database

Figs. 2-10 present the code used to create the database table which reflect **Fig. 1**. All code was run using VS Code with the MySQL extension, communicating with a local MySQL Community server on my computer.

```sql
Run
CREATE DATABASE PersonsDB;
-- Run queries on this database
Run
USE PersonsDB;

-- Create the persons table
Run | Select
CREATE TABLE Person(
    person_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    email VARCHAR(200) NOT NULL UNIQUE,
    dob DATE NOT NULL,
    -- Adding appropriate constraints
    -- Adapted from https://stackoverflow.com/questions/15560004/mysql-check-constraint-for-email-addresses
    CONSTRAINT email_check CHECK (email REGEXP '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$'),
    -- Constraint to check dob entered is not less than 01/01/1900, date in future is checked by trigger
    -- SQL handles dates in YYYY-mm-dd format
    CONSTRAINT dob_check CHECK (dob >= '1900-01-01')
);
```

Figure 2 Creating Person table

```sql
-- Create country table
Run | Select
CREATE TABLE Country(
    country_id INT PRIMARY KEY AUTO_INCREMENT,
    country_name VARCHAR(100) NOT NULL UNIQUE
);
```

Figure 3 Creating country table

```sql
-- Create city table
Run | Select
CREATE TABLE City(
    city_id INT PRIMARY KEY AUTO_INCREMENT,
    city VARCHAR(100) NOT NULL,
    country_id INT NOT NULL,
    FOREIGN KEY (country_id) REFERENCES Country(country_id)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
);
```

Figure 4 Creating city table

```sql
-- Create zip code table
▷Run | Select
CREATE TABLE ZipCode(
    zip_code_id INT PRIMARY KEY AUTO_INCREMENT,
    zip_code VARCHAR(9) NOT NULL,
    city_id INT NOT NULL,
    -- Setting references
    FOREIGN KEY (city_id) REFERENCES City(city_id)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
);
```

Figure 5 Creating zip code table

```sql
-- Create the neighbour table
-- Check constraint adapted from
-- https://stackoverflow.com/questions/15560004/mysql-check-constraint-for-email-addresses
▷Run | Select
CREATE TABLE Neighbour(
    neighbour_id INT PRIMARY KEY AUTO_INCREMENT,
    neighbour_first_name VARCHAR(200) NOT NULL,
    neighbour_last_name VARCHAR(200) NOT NULL,
    neighbour_email VARCHAR(200)  NOT NULL UNIQUE,
    CONSTRAINT neighbour_email_check CHECK (neighbour_email REGEXP '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$')
);
```

Figure 6 Creating neighbour table

```sql
-- Create address table
▷Run | Select
CREATE TABLE Address(
    address_id INT PRIMARY KEY AUTO_INCREMENT,
    street VARCHAR(200) NOT NULL,
    zip_code_id INT NOT NULL,
    neighbour_1_id INT NOT NULL,
    neighbour_2_id INT NOT NULL,
    FOREIGN KEY (zip_code_id) REFERENCES ZipCode(zip_code_id)
        ON DELETE RESTRICT
        ON UPDATE CASCADE,
    FOREIGN KEY (neighbour_1_id) REFERENCES Neighbour(neighbour_id)
        ON DELETE RESTRICT
        ON UPDATE CASCADE,
    FOREIGN KEY (neighbour_2_id) REFERENCES Neighbour(neighbour_id)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
);
```

Figure 7 Create address table

```sql
-- Create PersonAddress table
▷Run | Select
CREATE TABLE PersonAddress(
    person_id INT NOT NULL,
    address_id INT NOT NULL,
    -- Creating composite key
    PRIMARY KEY (person_id, address_id),
    -- Setting references
    FOREIGN KEY (person_id) REFERENCES Person(person_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (address_id) REFERENCES Address(address_id)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
);
```

Figure 8 Create person address table

```sql
-- Create Favourite table
▷Run | Select
CREATE TABLE Favourite(
    favourite_id INT PRIMARY KEY AUTO_INCREMENT,
    favourite_type VARCHAR(200) NOT NULL,
    favourite_name VARCHAR(200) NOT NULL UNIQUE
);
```

Figure 9 Create favourite table

```sql
-- Create PersonFavourite table
▷Run | Select
CREATE TABLE PersonFavourite(
    person_id INT NOT NULL,
    favourite_id INT NOT NULL,
    -- Creating composite key
    PRIMARY KEY (person_id, favourite_id),
    -- Setting references
    FOREIGN KEY (person_id) REFERENCES Person(person_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (favourite_id) REFERENCES Favourite(favourite_id)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
);
```

Figure 10 Create person favourite table

Fig. 11 presents the tables created from the above code.

Figure 11 All tables created

Triggers

I have setup triggers within my database to help with to improve data validation. I have two triggers **(Fig. 12)** for when a person is inserted/updated which will check their date of birth is not in the future, this couldn't be done using CONSTRAINT because it required a dynamic value (CURRENT_DATE()). **Fig. 13** shows the output of trying to insert a future DoB due to the trigger.

```sql
-- Trigger to check inputted dob is not in the future on insert
▷Run | Select
CREATE TRIGGER dob_check_future
-- Before person is inserted
BEFORE INSERT ON Person
-- For each person being inserted
FOR EACH ROW
BEGIN
    -- If the inputted dob is greater than the current date
    IF NEW.dob > CURRENT_DATE THEN
        -- This will throw an error and prevent the data being inserted
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Date of birth is in the future.';
    END IF;
END;

-- Trigger to check inputted dob is not in the future on update
▷Run | Select
CREATE TRIGGER dob_check_future_update
-- Before person is updated
BEFORE UPDATE ON Person
-- For each person being inserted
FOR EACH ROW
BEGIN
    -- If the inputted dob is greater than the current date
    IF NEW.dob > CURRENT_DATE THEN
        -- This will throw an error and prevent the data being inserted
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Date of birth is in the future.';
    END IF;
END;
```

Figure 12 Triggers to validate inserted DoB

```sql
▷Run | Select
INSERT INTO Person (first_name, last_name, email, dob)  Date of birth is in the future.
VALUES
('Person', '21', 'person21@email.com', '2025-03-15');  6ms      You, 22 minutes ago • Uncom
```

Figure 13 Inserting a future DoB

Scheduled Procedure

The database design can leave behind unused data, such as an addresses and favourites in use by no person(s). Because of this I created a procedure and used the event scheduler to trigger this procedure daily at midnight to clean up unused addresses, neighbours, and favourites. The benefit of this is running a small set of queries once daily instead of running multiple triggers after each delete/update. **Fig. 14** presents the code use to setup the scheduled procedure.

```sql
-- Procedure which will clean up unused data scheduled to run every day
-- Checking what favourites and addresses
-- are not part of a person relationship
-- Using a procedure as it is a convenient way to execute block of SQL
▷Run | Select
CREATE PROCEDURE RemoveUnusedData()
BEGIN

    -- Deleting favourites which do not belong to a person
    -- ON DELETE RESTRICT not triggered as no relationships
    DELETE FROM Favourite
    WHERE favourite_id
    NOT IN
    (SELECT DISTINCT favourite_id FROM PersonFavourite);

    -- Deleting addresses which do not belong to a person
    -- ON DELETE RESTRICT not triggered as no relationships
    DELETE FROM Address
    WHERE address_id
    NOT IN
    (SELECT DISTINCT address_id FROM PersonAddress);

    -- Delete unused neighbours
    DELETE FROM Neighbour
    WHERE neighbour_id
    NOT IN
    (SELECT DISTINCT neighbour_1_id FROM Address);

    DELETE FROM Neighbour
    WHERE neighbour_id
    NOT IN
    (SELECT DISTINCT neighbour_2_id FROM Address);

    -- Delete unused zip codes, cities, and countries
    DELETE FROM ZipCode
    WHERE zip_code NOT IN
    (SELECT DISTINCT zip_code FROM Address);

    DELETE FROM City
    WHERE city_id NOT IN
    (SELECT DISTINCT city_id FROM ZipCode);

    DELETE FROM Country
    WHERE country_id NOT IN
    (SELECT DISTINCT country_id FROM City);
END;

-- Enabling the event_scheduler
▷Run
SET GLOBAL event_scheduler = ON;

-- Adapted from:
-- https://stackoverflow.com/questions/9621355/how-to-schedule-a-mysql-query
-- Creating the event which will run the procedure to clean-up the data
-- Runs every day at midnight
▷Run | Select
CREATE EVENT DataClean
ON SCHEDULE EVERY 1 DAY
STARTS CURRENT_DATE
DO
BEGIN
    CALL RemoveUnusedData();
END;
```

Figure 14 Code for creating and scheduling the procedure to clean-up unused data

Inserting the Data

Figs. 15-25 present the code used to insert the case study data.

```sql
-- Insert Person data
▷ Run | Select
INSERT INTO Person (first_name, last_name, email, dob)
VALUES
('Person', '1', 'person1@email.com', '1995-03-15'),
('Person', '2', 'person2@email.com', '1993-06-22'),
('Person', '3', 'person3@email.com', '1991-09-10'),
('Person', '4', 'person4@email.com', '1998-12-05'),
('Person', '5', 'person5@email.com', '1983-11-30'),
('Person', '6', 'person6@email.com', '1989-07-18'),
('Person', '7', 'person7@email.com', '1996-04-25'),
('Person', '8', 'person8@email.com', '1990-01-09'),
('Person', '9', 'person9@email.com', '1993-08-17'),
('Person', '10', 'person10@email.com', '1997-10-22'),
('Person', '11', 'person11@email.com', '1992-05-13'),
('Person', '12', 'person12@email.com', '1986-02-27'),
('Person', '13', 'person13@email.com', '1991-11-25'),
('Person', '14', 'person14@email.com', '1987-02-01'),
('Person', '15', 'person15@email.com', '1984-08-12'),
('Person', '16', 'person16@email.com', '1990-03-09'),
('Person', '17', 'person17@email.com', '1995-11-17'),
('Person', '18', 'person18@email.com', '1994-06-20'),
('Person', '19', 'person19@email.com', '1992-12-11'),
('Person', '20', 'person20@email.com', '1988-09-25');
```

Figure 15 Insert person data

```sql
-- Insert Country data
▷ Run | Select
INSERT INTO Country (country_name)
VALUES
('England'),
('Scotland'),
('Wales');
```

Figure 16 Insert country data

```sql
-- Insert City data
-- Insert country ID by selecting on country name
▷ Run | Select
INSERT INTO City (city, country_id)
VALUES
('London', (SELECT country_id FROM Country WHERE country_name = 'England')),
('Manchester', (SELECT country_id FROM Country WHERE country_name = 'England')),
('Birmingham', (SELECT country_id FROM Country WHERE country_name = 'England')),
('Edinburgh', (SELECT country_id FROM Country WHERE country_name = 'Scotland')),
('Bristol', (SELECT country_id FROM Country WHERE country_name = 'England')),
('Liverpool', (SELECT country_id FROM Country WHERE country_name = 'England')),
('Glasgow', (SELECT country_id FROM Country WHERE country_name = 'Scotland')),
('Leeds', (SELECT country_id FROM Country WHERE country_name = 'England')),
('Newcastle', (SELECT country_id FROM Country WHERE country_name = 'England')),
('Cardiff', (SELECT country_id FROM Country WHERE country_name = 'Wales')),
('Sheffield', (SELECT country_id FROM Country WHERE country_name = 'England')),
('Nottingham', (SELECT country_id FROM Country WHERE country_name = 'England')),
('Cambridge', (SELECT country_id FROM Country WHERE country_name = 'England')),
('Oxford', (SELECT country_id FROM Country WHERE country_name = 'England')),
('Southampton', (SELECT country_id FROM Country WHERE country_name = 'England')),
('Leicester', (SELECT country_id FROM Country WHERE country_name = 'England')),
('Norwich', (SELECT country_id FROM Country WHERE country_name = 'England'));
```

Figure 17 Insert city data

```sql
-- Insert Zip Code data
-- Insert city ID by selecting on city name
▷ Run | Select
INSERT INTO ZipCode (zip_code, city_id)
VALUES
('E1 6AN', (SELECT city_id FROM City WHERE city = 'London')),
('M1 2WD', (SELECT city_id FROM City WHERE city = 'Manchester')),
('CF10 3BC', (SELECT city_id FROM City WHERE city = 'Cardiff')),
('B1 1AB', (SELECT city_id FROM City WHERE city = 'Birmingham')),
('EH1 1YZ', (SELECT city_id FROM City WHERE city = 'Edinburgh')),
('BS1 3XE', (SELECT city_id FROM City WHERE city = 'Bristol')),
('L1 1AA', (SELECT city_id FROM City WHERE city = 'Liverpool')),
('G1 2TF', (SELECT city_id FROM City WHERE city = 'Glasgow')),
('LS1 3AB', (SELECT city_id FROM City WHERE city = 'Leeds')),
('NE1 2AB', (SELECT city_id FROM City WHERE city = 'Newcastle')),
('CF10 3AF', (SELECT city_id FROM City WHERE city = 'Cardiff')),
('S1 4GT', (SELECT city_id FROM City WHERE city = 'Sheffield')),
('NG1 2PB', (SELECT city_id FROM City WHERE city = 'Nottingham')),
('EH1 1AB', (SELECT city_id FROM City WHERE city = 'Edinburgh')),
('CF10 2NF', (SELECT city_id FROM City WHERE city = 'Cardiff')),
('CB1 2SE', (SELECT city_id FROM City WHERE city = 'Cambridge')),
('OX2 6TP', (SELECT city_id FROM City WHERE city = 'Oxford')),
('SO14 3HL', (SELECT city_id FROM City WHERE city = 'Southampton')),
('LE1 3PL', (SELECT city_id FROM City WHERE city = 'Leicester')),
('NR1 4BE', (SELECT city_id FROM City WHERE city = 'Norwich'));
```

Figure 18 Insert zip code data

```sql
-- Insert Neighbour data
▷Run | Select
INSERT INTO Neighbour(neighbour_first_name, neighbour_last_name, neighbour_email)
VALUES
('Neighbor', 'A', 'neighborA@email.com'),
('Neighbor', 'B', 'neighborB@email.com'),
('Neighbor', 'C', 'neighborC@email.com'),
('Neighbor', 'D', 'neighborD@email.com'),
('Neighbor', 'E', 'neighborE@email.com'),
('Neighbor', 'F', 'neighborF@email.com'),
('Neighbor', 'G', 'neighborG@email.com'),
('Neighbor', 'H', 'neighborH@email.com'),
('Neighbor', 'I', 'neighborI@email.com'),
('Neighbor', 'J', 'neighborJ@email.com'),
('Neighbor', 'K', 'neighborK@email.com'),
('Neighbor', 'L', 'neighborL@email.com'),
('Neighbor', 'M', 'neighborM@email.com'),
('Neighbor', 'N', 'neighborN@email.com'),
('Neighbor', 'O', 'neighborO@email.com'),
('Neighbor', 'P', 'neighborP@email.com'),
('Neighbor', 'Q', 'neighborQ@email.com'),
('Neighbor', 'R', 'neighborR@email.com'),
('Neighbor', 'S', 'neighborS@email.com'),
('Neighbor', 'T', 'neighborT@email.com'),
('Neighbor', 'U', 'neighborU@email.com'),
('Neighbor', 'V', 'neighborV@email.com'),
('Neighbor', 'W', 'neighborW@email.com'),
('Neighbor', 'X', 'neighborX@email.com'),
('Neighbor', 'Y', 'neighborY@email.com'),
('Neighbor', 'Z', 'neighborZ@email.com'),
('Neighbor', 'AA', 'neighborAA@email.com'),
('Neighbor', 'AB', 'neighborAB@email.com'),
('Neighbor', 'AC', 'neighborAC@email.com'),
('Neighbor', 'AD', 'neighborAD@email.com'),
('Neighbor', 'AE', 'neighborAE@email.com'),
('Neighbor', 'AF', 'neighborAF@email.com'),
('Neighbor', 'AG', 'neighborAG@email.com'),
('Neighbor', 'AH', 'neighborAH@email.com'),
('Neighbor', 'AI', 'neighborAI@email.com'),
('Neighbor', 'AJ', 'neighborAJ@email.com'),
('Neighbor', 'AK', 'neighborAK@email.com'),
('Neighbor', 'AL', 'neighborAL@email.com'),
('Neighbor', 'AM', 'neighborAM@email.com'),
('Neighbor', 'AN', 'neighborAN@email.com');
```

Figure 19 Insert neighbour data

```sql
-- Insert Address data with reference to ZipCode (considering both zip_code and city) and Neighbours
Run | Select
INSERT INTO Address(street, zip_code_id, neighbour_1_id, neighbour_2_id)
VALUES
('12 Maple St',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'E1 6AN'
     AND city_id = (SELECT city_id FROM City WHERE city = 'London')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborA@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborB@email.com')
),
('45 Oak Ave',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'M1 2WD'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Manchester')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborC@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborD@email.com')
),
('89 Pine Rd',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'B1 1AB'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Birmingham')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborE@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborF@email.com')
),
('23 Birch St',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'EH1 1YZ'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Edinburgh')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborG@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborH@email.com')
),      You, 8 hours ago • Updated task 1 code
('67 Cedar Ln',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'BS1 3XE'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Bristol')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborI@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborJ@email.com')
),
('56 Elm St',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'L1 1AA'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Liverpool')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborK@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborL@email.com')
),
('12 Maple St',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'G1 2TF'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Glasgow')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborM@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborN@email.com')
),
('89 Oak Dr',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'LS1 3AB'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Leeds')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborO@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborP@email.com')
),
('123 Pine Rd',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'NE1 2AB'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Newcastle')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborQ@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborR@email.com')
),
('15 Elm St',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'CF10 3AF'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Cardiff')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborS@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborT@email.com')
),
```

Figure 20 Insert address data (1)

```
('78 Oak Ln',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'S1 4GT'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Sheffield')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborU@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborV@email.com')
),
('56 Birch Rd',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'NG1 2PB'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Nottingham')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborW@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborX@email.com')
),
('10 Holy St',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'CF10 2NF'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Cardiff')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborY@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborZ@email.com')
),
('34 Willow Rd',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'EH1 1AB'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Edinburgh')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborAA@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborAB@email.com')
),
('78 Cedar Ave',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'CB1 2SE'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Cambridge')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborAC@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborAD@email.com')
),
('45 Maple Rd',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'OX2 6TP'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Oxford')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborAE@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborAF@email.com')
),
('23 Birch Ave',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'SO14 3HL'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Southampton')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborAG@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborAH@email.com')
),
('12 Elm Blvd',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'LE1 3PL'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Leicester')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborAI@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborAJ@email.com')
),
('56 Oak Rd',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'NR1 4BE'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Norwich')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborAK@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborAL@email.com')
),
('89 Pine Ave',
    (SELECT zip_code_id FROM ZipCode
     WHERE zip_code = 'CF10 3BC'
     AND city_id = (SELECT city_id FROM City WHERE city = 'Cardiff')),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborAM@email.com'),
    (SELECT neighbour_id FROM Neighbour WHERE neighbour_email = 'neighborAN@email.com')
);
```

Figure 21 Insert address data (2)

```sql
-- Insert PersonAddress data
-- Insert person ID by person email
-- Insert address ID by selecting on street and zip code
▷ Run | Select
INSERT INTO PersonAddress (person_id, address_id)
VALUES
(
  (SELECT person_id FROM Person WHERE email = 'person1@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'E1 6AN'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person2@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'M1 2WD'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person3@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'B1 1AB'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person4@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'EH1 1YZ'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person5@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'BS1 3XE'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person6@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'L1 1AA'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person7@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'G1 2TF'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person8@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'LS1 3AB'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person9@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'NE1 2AB'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person10@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'CF10 3AF'))
),
```

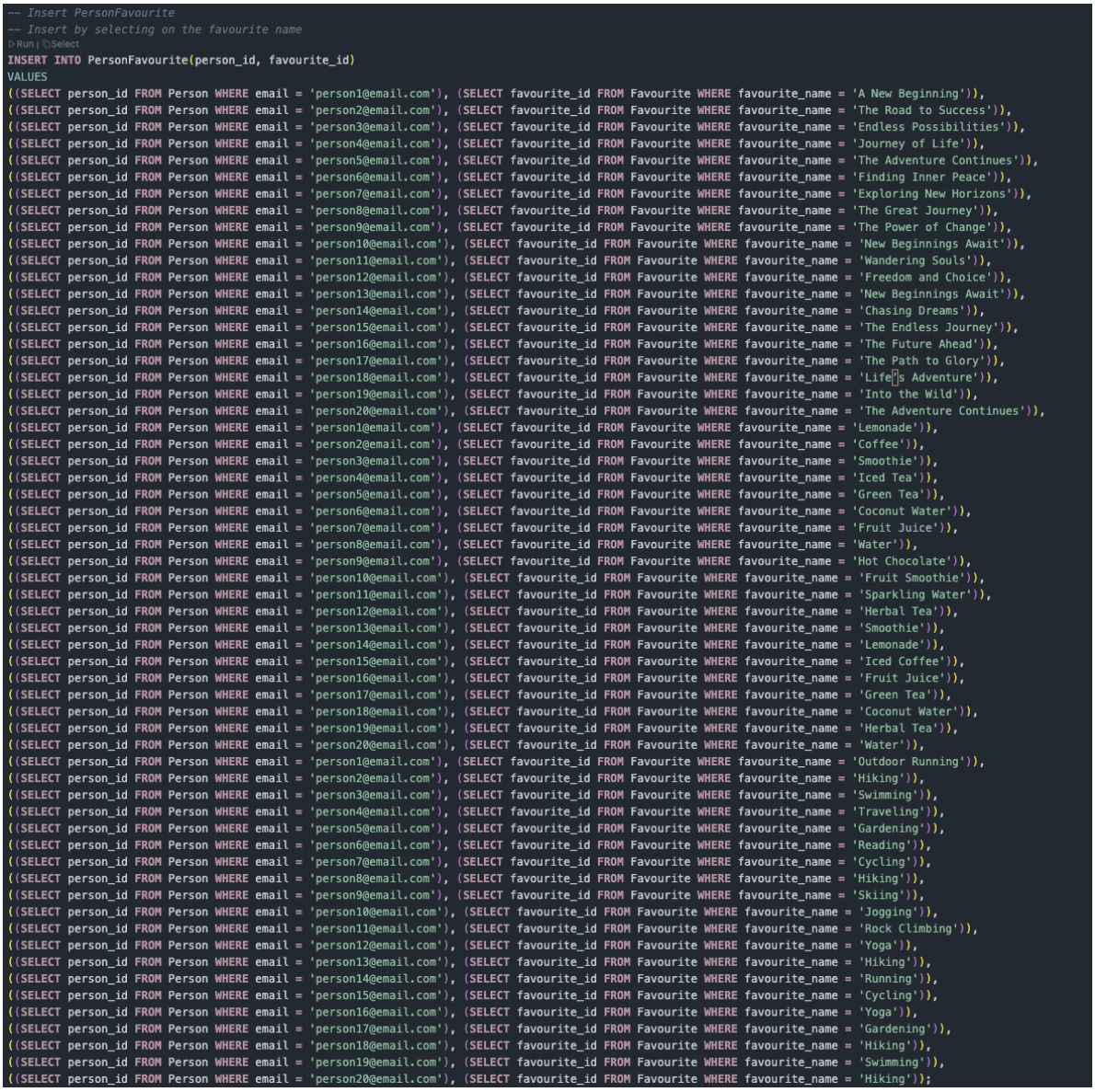
Figure 22 Insert person address data (1)

```
(
  (SELECT person_id FROM Person WHERE email = 'person10@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'CF10 3AF'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person11@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'S1 4GT'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person12@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'NG1 2PB'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person13@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'CF10 2NF'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person14@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'EH1 1AB'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person15@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'CB1 2SE'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person16@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'OX2 6TP'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person17@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'SO14 3HL'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person18@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'LE1 3PL'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person19@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'NR1 4BE'))
),
(
  (SELECT person_id FROM Person WHERE email = 'person20@email.com'),
  (SELECT address_id FROM Address WHERE zip_code_id = (SELECT zip_code_id FROM ZipCode WHERE zip_code = 'CF10 3BC'))
);
```

Figure 23 Insert person address data (2)

```
-- Insert Favourites
▷Run | Select
INSERT INTO Favourite(favourite_type, favourite_name)
VALUES
('Book', 'A New Beginning'),
('Book', 'The Road to Success'),
('Book', 'Endless Possibilities'),
('Book', 'Journey of Life'),
('Book', 'The Adventure Continues'),
('Book', 'Finding Inner Peace'),
('Book', 'Exploring New Horizons'),
('Book', 'The Great Journey'),
('Book', 'The Power of Change'),
('Book', 'New Beginnings Await'),
('Book', 'Wandering Souls'),
('Book', 'Freedom and Choice'),
('Book', 'Chasing Dreams'),
('Book', 'The Endless Journey'),
('Book', 'The Future Ahead'),
('Book', 'The Path to Glory'),
('Book', 'Life's Adventure'),
('Book', 'Into the Wild'),
('Drink', 'Lemonade'),
('Drink', 'Coffee'),
('Drink', 'Smoothie'),
('Drink', 'Iced Tea'),
('Drink', 'Green Tea'),
('Drink', 'Coconut Water'),
('Drink', 'Fruit Juice'),
('Drink', 'Water'),
('Drink', 'Hot Chocolate'),
('Drink', 'Fruit Smoothie'),
('Drink', 'Sparkling Water'),
('Drink', 'Herbal Tea'),
('Drink', 'Iced Coffee'),
('Activity', 'Outdoor Running'),
('Activity', 'Hiking'),
('Activity', 'Swimming'),
('Activity', 'Traveling'),
('Activity', 'Gardening'),
('Activity', 'Reading'),
('Activity', 'Cycling'),
('Activity', 'Skiing'),
('Activity', 'Jogging'),
('Activity', 'Rock Climbing'),
('Activity', 'Yoga'),
('Activity', 'Running');
```

Figure 24 Insert favourite data



```sql
-- Insert PersonFavourite
-- Insert by selecting on the favourite name
▷Run | ⊡Select
INSERT INTO PersonFavourite(person_id, favourite_id)
VALUES
((SELECT person_id FROM Person WHERE email = 'person1@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'A New Beginning')),
((SELECT person_id FROM Person WHERE email = 'person2@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'The Road to Success')),
((SELECT person_id FROM Person WHERE email = 'person3@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Endless Possibilities')),
((SELECT person_id FROM Person WHERE email = 'person4@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Journey of Life')),
((SELECT person_id FROM Person WHERE email = 'person5@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'The Adventure Continues')),
((SELECT person_id FROM Person WHERE email = 'person6@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Finding Inner Peace')),
((SELECT person_id FROM Person WHERE email = 'person7@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Exploring New Horizons')),
((SELECT person_id FROM Person WHERE email = 'person8@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'The Great Journey')),
((SELECT person_id FROM Person WHERE email = 'person9@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'The Power of Change')),
((SELECT person_id FROM Person WHERE email = 'person10@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'New Beginnings Await')),
((SELECT person_id FROM Person WHERE email = 'person11@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Wandering Souls')),
((SELECT person_id FROM Person WHERE email = 'person12@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Freedom and Choice')),
((SELECT person_id FROM Person WHERE email = 'person13@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'New Beginnings Await')),
((SELECT person_id FROM Person WHERE email = 'person14@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Chasing Dreams')),
((SELECT person_id FROM Person WHERE email = 'person15@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'The Endless Journey')),
((SELECT person_id FROM Person WHERE email = 'person16@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'The Future Ahead')),
((SELECT person_id FROM Person WHERE email = 'person17@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'The Path to Glory')),
((SELECT person_id FROM Person WHERE email = 'person18@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Life□s Adventure')),
((SELECT person_id FROM Person WHERE email = 'person19@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Into the Wild')),
((SELECT person_id FROM Person WHERE email = 'person20@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'The Adventure Continues')),
((SELECT person_id FROM Person WHERE email = 'person1@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Lemonade')),
((SELECT person_id FROM Person WHERE email = 'person2@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Coffee')),
((SELECT person_id FROM Person WHERE email = 'person3@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Smoothie')),
((SELECT person_id FROM Person WHERE email = 'person4@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Iced Tea')),
((SELECT person_id FROM Person WHERE email = 'person5@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Green Tea')),
((SELECT person_id FROM Person WHERE email = 'person6@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Coconut Water')),
((SELECT person_id FROM Person WHERE email = 'person7@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Fruit Juice')),
((SELECT person_id FROM Person WHERE email = 'person8@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Water')),
((SELECT person_id FROM Person WHERE email = 'person9@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Hot Chocolate')),
((SELECT person_id FROM Person WHERE email = 'person10@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Fruit Smoothie')),
((SELECT person_id FROM Person WHERE email = 'person11@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Sparkling Water')),
((SELECT person_id FROM Person WHERE email = 'person12@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Herbal Tea')),
((SELECT person_id FROM Person WHERE email = 'person13@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Smoothie')),
((SELECT person_id FROM Person WHERE email = 'person14@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Lemonade')),
((SELECT person_id FROM Person WHERE email = 'person15@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Iced Coffee')),
((SELECT person_id FROM Person WHERE email = 'person16@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Fruit Juice')),
((SELECT person_id FROM Person WHERE email = 'person17@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Green Tea')),
((SELECT person_id FROM Person WHERE email = 'person18@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Coconut Water')),
((SELECT person_id FROM Person WHERE email = 'person19@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Herbal Tea')),
((SELECT person_id FROM Person WHERE email = 'person20@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Water')),
((SELECT person_id FROM Person WHERE email = 'person1@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Outdoor Running')),
((SELECT person_id FROM Person WHERE email = 'person2@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Hiking')),
((SELECT person_id FROM Person WHERE email = 'person3@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Swimming')),
((SELECT person_id FROM Person WHERE email = 'person4@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Traveling')),
((SELECT person_id FROM Person WHERE email = 'person5@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Gardening')),
((SELECT person_id FROM Person WHERE email = 'person6@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Reading')),
((SELECT person_id FROM Person WHERE email = 'person7@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Cycling')),
((SELECT person_id FROM Person WHERE email = 'person8@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Hiking')),
((SELECT person_id FROM Person WHERE email = 'person9@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Skiing')),
((SELECT person_id FROM Person WHERE email = 'person10@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Jogging')),
((SELECT person_id FROM Person WHERE email = 'person11@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Rock Climbing')),
((SELECT person_id FROM Person WHERE email = 'person12@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Yoga')),
((SELECT person_id FROM Person WHERE email = 'person13@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Hiking')),
((SELECT person_id FROM Person WHERE email = 'person14@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Running')),
((SELECT person_id FROM Person WHERE email = 'person15@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Cycling')),
((SELECT person_id FROM Person WHERE email = 'person16@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Yoga')),
((SELECT person_id FROM Person WHERE email = 'person17@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Gardening')),
((SELECT person_id FROM Person WHERE email = 'person18@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Hiking')),
((SELECT person_id FROM Person WHERE email = 'person19@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Swimming')),
((SELECT person_id FROM Person WHERE email = 'person20@email.com'), (SELECT favourite_id FROM Favourite WHERE favourite_name = 'Hiking'));
```

Figure 25 Insert person favourite data

Figs. 26-35 show the data as it has been inserted into the tables.

person_id int	first_name varchar(100)	last_name varchar(100)	email varchar(200)	dob date
1	Person	1	person1@email.com	1995-03-15
2	Person	2	person2@email.com	1993-06-22
3	Person	3	person3@email.com	1991-09-10
4	Person	4	person4@email.com	1998-12-05
5	Person	5	person5@email.com	1983-11-30
6	Person	6	person6@email.com	1989-07-18
7	Person	7	person7@email.com	1996-04-25
8	Person	8	person8@email.com	1990-01-09
9	Person	9	person9@email.com	1993-08-17
10	Person	10	person10@email.com	1997-10-22
11	Person	11	person11@email.com	1992-05-13
12	Person	12	person12@email.com	1986-02-27
13	Person	13	person13@email.com	1991-11-25
14	Person	14	person14@email.com	1987-02-01
15	Person	15	person15@email.com	1984-08-12
16	Person	16	person16@email.com	1990-03-09
17	Person	17	person17@email.com	1995-11-17
18	Person	18	person18@email.com	1994-06-20
19	Person	19	person19@email.com	1992-12-11
20	Person	20	person20@email.com	1988-09-25

Figure 26 Inserted person data

country_id int	country_name varchar(100)
1	England
2	Scotland
3	Wales

Figure 27 Inserted country data

city_id int	city varchar(100)	country_id int
1	London	1
2	Manchester	1
3	Birmingham	1
4	Edinburgh	2
5	Bristol	1
6	Liverpool	1
7	Glasgow	2
8	Leeds	1
9	Newcastle	1
10	Cardiff	3
11	Sheffield	1
12	Nottingham	1
13	Cambridge	1
14	Oxford	1
15	Southampton	1
16	Leicester	1
17	Norwich	1

Figure 28 Inserted city data

zip_code_id int	zip_code varchar(9)	city_id int
1	E1 6AN	1
2	M1 2WD	2
3	CF10 3BC	10
4	B1 1AB	3
5	EH1 1YZ	4
6	BS1 3XE	5
7	L1 1AA	6
8	G1 2TF	7
9	LS1 3AB	8
10	NE1 2AB	9
11	CF10 3AF	10
12	S1 4GT	11
13	NG1 2PB	12
14	EH1 1AB	4
15	CF10 2NF	10
16	CB1 2SE	13
17	OX2 6TP	14
18	SO14 3HL	15
19	LE1 3PL	16
20	NR1 4BE	17

Figure 29 Inserted zip code data

neighbour_id int	neighbour_first_name varchar(200)	neighbour_last_name varchar(200)	neighbour_email varchar(200)
1	Neighbor	A	neighborA@email.com
2	Neighbor	B	neighborB@email.com
3	Neighbor	C	neighborC@email.com
4	Neighbor	D	neighborD@email.com
5	Neighbor	E	neighborE@email.com
6	Neighbor	F	neighborF@email.com
7	Neighbor	G	neighborG@email.com
8	Neighbor	H	neighborH@email.com
9	Neighbor	I	neighborI@email.com
10	Neighbor	J	neighborJ@email.com
11	Neighbor	K	neighborK@email.com
12	Neighbor	L	neighborL@email.com
13	Neighbor	M	neighborM@email.com
14	Neighbor	N	neighborN@email.com
15	Neighbor	O	neighborO@email.com
16	Neighbor	P	neighborP@email.com
17	Neighbor	Q	neighborQ@email.com
18	Neighbor	R	neighborR@email.com
19	Neighbor	S	neighborS@email.com
20	Neighbor	T	neighborT@email.com
21	Neighbor	U	neighborU@email.com
22	Neighbor	V	neighborV@email.com
23	Neighbor	W	neighborW@email.com
24	Neighbor	X	neighborX@email.com
25	Neighbor	Y	neighborY@email.com
26	Neighbor	Z	neighborZ@email.com
27	Neighbor	AA	neighborAA@email.com
28	Neighbor	AB	neighborAB@email.com
29	Neighbor	AC	neighborAC@email.com
30	Neighbor	AD	neighborAD@email.com
31	Neighbor	AE	neighborAE@email.com
32	Neighbor	AF	neighborAF@email.com
33	Neighbor	AG	neighborAG@email.com
34	Neighbor	AH	neighborAH@email.com
35	Neighbor	AI	neighborAI@email.com
36	Neighbor	AJ	neighborAJ@email.com
37	Neighbor	AK	neighborAK@email.com
38	Neighbor	AL	neighborAL@email.com
39	Neighbor	AM	neighborAM@email.com
40	Neighbor	AN	neighborAN@email.com

Figure 30 Inserted neighbour data

address_id int	street varchar(200)	zip_code_id int	neighbour_1_id int	neighbour_2_id int
1	12 Maple St	1	1	2
2	45 Oak Ave	2	3	4
3	89 Pine Rd	4	5	6
4	23 Birch St	5	7	8
5	67 Cedar Ln	6	9	10
6	56 Elm St	7	11	12
7	12 Maple St	8	13	14
8	89 Oak Dr	9	15	16
9	123 Pine Rd	10	17	18
10	15 Elm St	11	19	20
11	78 Oak Ln	12	21	22
12	56 Birch Rd	13	23	24
13	10 Holy St	15	25	26
14	34 Willow Rd	14	27	28
15	78 Cedar Ave	16	29	30
16	45 Maple Rd	17	31	32
17	23 Birch Ave	18	33	34
18	12 Elm Blvd	19	35	36
19	56 Oak Rd	20	37	38
20	89 Pine Ave	3	39	40

Figure 31 Inserted address data

person_id int	address_id int
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20

Figure 32 Inserted person address data

favourite_id int	favourite_type varchar(200)	favourite_name varchar(200)
1	Book	A New Beginning
2	Book	The Road to Success
3	Book	Endless Possibilities
4	Book	Journey of Life
5	Book	The Adventure Continues
6	Book	Finding Inner Peace
7	Book	Exploring New Horizons
8	Book	The Great Journey
9	Book	The Power of Change
10	Book	New Beginnings Await
11	Book	Wandering Souls
12	Book	Freedom and Choice
13	Book	Chasing Dreams
14	Book	The Endless Journey
15	Book	The Future Ahead
16	Book	The Path to Glory
17	Book	Life's Adventure
18	Book	Into the Wild
19	Drink	Lemonade
20	Drink	Coffee
21	Drink	Smoothie
22	Drink	Iced Tea
23	Drink	Green Tea
24	Drink	Coconut Water
25	Drink	Fruit Juice
26	Drink	Water
27	Drink	Hot Chocolate
28	Drink	Fruit Smoothie
29	Drink	Sparkling Water
30	Drink	Herbal Tea
31	Drink	Iced Coffee
32	Activity	Outdoor Running
33	Activity	Hiking
34	Activity	Swimming
35	Activity	Traveling
36	Activity	Gardening
37	Activity	Reading
38	Activity	Cycling
39	Activity	Skiing
40	Activity	Jogging
41	Activity	Rock Climbing
42	Activity	Yoga
43	Activity	Running

Figure 33 Inserted favourite data

person_id int	favourite_id int
1	1
2	2
3	3
4	4
5	5
20	5
6	6
7	7
8	8
9	9
10	10
13	10
11	11
12	12
14	13
15	14
16	15
17	16
18	17
19	18
1	19
14	19
2	20
3	21
13	21
4	22
5	23
17	23
6	24
18	24
7	25
16	25
8	26
20	26
9	27
10	28
11	29
12	30
19	30
15	31
1	32
2	33
8	33
13	33
18	33

Figure 34 Inserted person favourite data (1)

20	33
3	34
19	34
4	35
5	36
17	36
6	37
7	38
15	38
9	39
10	40
11	41
12	42
16	42
14	43

Figure 35 Inserted person favourite data (2)

SQL Workbench Reverse Engineered ERD

Fig. 36 shows the reverse engineered ERD diagram generated from SQL Workbench, showcasing my final database implementation.

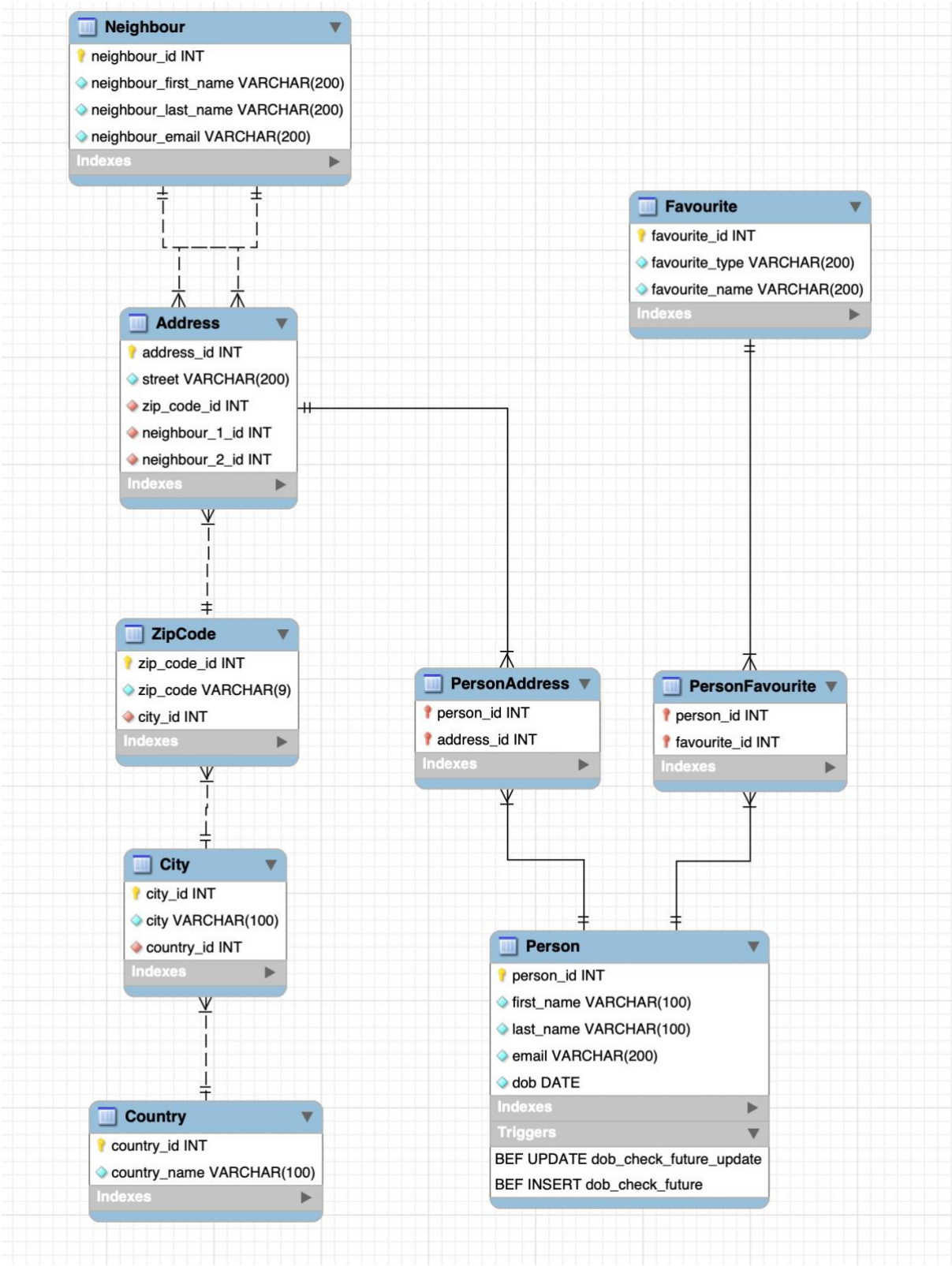


Figure 36 Reverse-engineered ERD using SQLWorkbench

Queries & Results

Context behind the design of my queries is included in SQL comments within the screenshots.

1) Display person's name and their age in years

```sql
-- Query 1
-- TIMESTAMPDIFF() recommended by MySQL https://dev.mysql.com/doc/refman/8.4/en/date-calculations.html
-- for being optimal
▷ Run | +Tab | JSON | Select
SELECT
p.first_name, p.last_name,
TIMESTAMPDIFF(YEAR, p.dob, CURRENT_DATE()) AS age
FROM Person p;
```

Figure 37 Query 1 SQL

first_name varchar(100)	last_name varchar(100)	age
Person	1	30
Person	2	31
Person	3	33
Person	4	26
Person	5	41
Person	6	35
Person	7	28
Person	8	35
Person	9	31
Person	10	27
Person	11	32
Person	12	39
Person	13	33
Person	14	38
Person	15	40
Person	16	35
Person	17	29
Person	18	30
Person	19	32
Person	20	36

Figure 38 Query 1 Result

2) Group Persons by their favourite drink and return average age of each group

```sql
-- Query 2
-- Using AVG on the age calculation used in query 1
-- Using ROUND on the calculated AVG to limit trailing zeros
▷Run | +Tab | JSON | Select
SELECT
    f.favourite_name AS drink_name,
    ROUND(AVG(TIMESTAMPDIFF(YEAR, p.dob, CURRENT_DATE())), 1) AS average_age
FROM PersonFavourite pf
-- Join Person with PersonFavourite on person_id
JOIN Person p ON pf.person_id = p.person_id
-- Join Favourite table to get the drink information
JOIN Favourite f ON pf.favourite_id = f.favourite_id
-- Filter the results by only the favourites of type drink
WHERE f.favourite_type = 'Drink'
-- Group by drink name for average per drink
GROUP BY f.favourite_name;
```

Figure 39 Query 2 SQL

drink_name varchar	average_age decimal
Coconut Water	32.5
Coffee	31.0
Fruit Juice	31.5
Fruit Smoothie	27.0
Green Tea	35.0
Herbal Tea	35.5
Hot Chocolate	31.0
Iced Coffee	40.0
Iced Tea	26.0
Lemonade	34.0
Smoothie	33.0
Sparkling Water	32.0
Water	35.5

Figure 40 Query 2 result

3) Display average age of people who likes Hiking

```sql
-- Query 3
-- Using AVG on the age calculation used in query 1/2
-- Using ROUND on the calculated AVG to limit trailing zeros
▷ Run | ⇥ Tab | JSON | Select
SELECT
    f.favourite_name AS activity_name,
    ROUND(AVG(TIMESTAMPDIFF(YEAR, p.dob, CURRENT_DATE())), 1) AS average_age
FROM PersonFavourite pf
-- Join Person with PersonFavourite on person_id
JOIN Person p ON pf.person_id = p.person_id
-- Join Favourite with PersonFavourite on favourite_id
-- Needed to get favourite details
JOIN Favourite f ON pf.favourite_id = f.favourite_id
-- Filter by activity type and activity name (only hiking)
WHERE f.favourite_type = 'Activity' AND f.favourite_name = 'Hiking'
-- Group by hiking to return a single average
GROUP BY f.favourite_name;
```

Figure 41 Query 3 SQL

activity_name varchar	average_age decimal
Hiking	33.0

Figure 42 Query 3 result

4) Display the total number of people from each City and sort it in ascending order by total number of people

```sql
-- Query 4
-- COUNT(DISTINCT pa.person_id) ensures person with multiple addresses in same city only counts
-- once in population sum
▷Run | +Tab | JSON | Select
SELECT
    c.city,
    co.country_name AS country,
    COUNT(DISTINCT pa.person_id) AS population
FROM PersonAddress pa
-- Join Address with PersonAddress on address_id
JOIN Address a ON pa.address_id = a.address_id
-- Join ZipCode with Address on address_id
JOIN ZipCode z ON a.zip_code_id = z.zip_code_id
-- Join City with ZipCode on zip_code_id
JOIN City c ON z.city_id = c.city_id
-- Join Country with City on city_id
JOIN Country co ON c.country_id = co.country_id
GROUP BY c.city, co.country_name
ORDER BY population ASC;
```

Figure 43 Query 4 SQL

city varchar	country varchar	population bigint
Birmingham	England	1
Bristol	England	1
Cambridge	England	1
Glasgow	Scotland	1
Leeds	England	1
Leicester	England	1
Liverpool	England	1
London	England	1
Manchester	England	1
Newcastle	England	1
Norwich	England	1
Nottingham	England	1
Oxford	England	1
Sheffield	England	1
Southampton	England	1
Edinburgh	Scotland	2
Cardiff	Wales	3

Figure 44 Query 4 result

5) Display name of person(s) whose neighbour is neighbour C

```sql
-- Query 5
▷ Run | + Tab | JSON | Select
SELECT
    p.first_name,
    p.last_name
FROM Person p
-- Join PersonAddress with Person on person_id
JOIN PersonAddress pa ON p.person_id = pa.person_id
-- Join Address with PersonAddress on address_id
JOIN Address a ON pa.address_id = a.address_id
-- Join Neighbour with Address on both neighbours
JOIN Neighbour n1 ON a.neighbour_1_id = n1.neighbour_id
JOIN Neighbour n2 ON a.neighbour_2_id = n2.neighbour_id
-- Filter by the neighbour name 'Neighbor C'
-- OR to check both neighbour 1 and 2
WHERE
    (n1.neighbour_first_name = "Neighbor"
    AND n1.neighbour_last_name = "C")
    OR
    (n2.neighbour_first_name = "Neighbor"
    AND n2.neighbour_last_name = "C");
```

Figure 45 Query 5 SQL

first_name varchar	last_name varchar
Person	2

Figure 46 Query 5 result

References

- Datacamp (2024) ***What is Third Normal Form (3NF)?*** Available from: https://www.datacamp.com/tutorial/third-normal-form [Accessed February 10 2025]