

ABERTAWE LEISURE CENTRE
(ALC)

Table of Contents

1. Database Creation	3
References.....	9

1. Database Creation

```
-----DATABASE CREATION-----  
  
-----TABLES-----  
  
CREATE SEQUENCE mem_id_seq;                                -- Sequence for member id  
SELECT setval('mem_id_seq',10001);  
  
CREATE TABLE member_details  
(  
    member_id      TEXT DEFAULT ('MEM'||nextval('mem_id_seq')) NOT NULL PRIMARY KEY,  
    first_name     VARCHAR(50) NOT NULL,  
    last_name      VARCHAR(50),  
    DOB            DATE,  
    gender          VARCHAR(10),  
    address         VARCHAR(100),  
    phone_no       VARCHAR(11) NOT NULL,  
    date_joined    DATE,  
  
    CHECK (gender in ('male', 'female', 'other'))  
);
```

Figure 1: creating tables

(Source: screenshot from “postgresql Pgadmin4”)

The above table stores details of the new member there is a sequence created for the member ID which starts with ”MEM” and ends with a 5 digit number. the columns names represent what they stand for the name consists of two parts which is first name and the last name which makes querying easy for us since the index will take much less space than intended and we have added a check constraint for that column gender which allows us to restrict information which is out of bounds.

```

insert into member_details (first_name, last_name, gender, DOB, date_joined, address, phone_no) values
insert into member_details (first_name, last_name, gender, DOB, date_joined, address, phone_no) values
insert into member_details (first_name, last_name, gender, DOB, date_joined, address, phone_no) values
insert into member_details (first_name, last_name, gender, DOB, date_joined, address, phone_no) values
insert into member_details (first_name, last_name, gender, DOB, date_joined, address, phone_no) values
insert into member_details (first_name, last_name, gender, DOB, date_joined, address, phone_no) values
insert into member_details (first_name, last_name, gender, DOB, date_joined, address, phone_no) values
insert into member_details (first_name, last_name, gender, DOB, date_joined, address, phone_no) values
insert into member_details (first_name, last_name, gender, DOB, date_joined, address, phone_no) values

values ('Marijo', 'Lovatt', 'female', '1990/12/16', '1999/04/16', '15th Floor', '7557179186');
values ('Candy', 'Griffith', 'female', '2001/04/19', '1997/04/02', 'Suite 58', '1595237513');
values ('Saunderson', 'Chastney', 'male', '1998/01/10', '1998/07/12', 'Apt 905', '9041498334');
values ('Nil', 'Earwicker', 'male', '1995/02/09', '2002/07/09', 'PO Box 71030', '4658224927');
values ('Brett', 'Vernalls', 'male', '2001/04/11', '2001/11/12', '9th Floor', '9552200344');
values ('Murray', 'Tansey', 'male', '1992/02/18', '1992/09/17', 'PO Box 50323', '5894729362');
values ('Eb', 'Monnery', 'male', '2000/06/07', '1999/09/20', '16th Floor', '5463122701');
values ('Amitie', 'Calton', 'female', '1995/04/22', '2002/12/26', '4th Floor', '4502867703');
values ('Delilah', 'Gavaghan', 'female', '1993/01/24', '2001/09/18', '20th Floor', '6323803078');
values ('Marleah', 'Stork', 'female', '1995/07/06', '1998/06/29', 'Suite 60', '6764682747');

```

Figure 2

(Source: screenshot from “postgresql Pgadmin4”)

now we are inserting values into the first table which is member_details. as we insert values into the table the member ID is created through an auto sequence and which is also a primary key and represents the whole table.

```

25 CREATE SEQUENCE inst_id_seq;           -- Sequence for instructor id
26 SELECT setval('inst_id_seq',101);
27
28 CREATE TABLE instructor_details
29 (
30     instructor_id      TEXT DEFAULT ('MEM'||nextval('inst_id_seq')) PRIMARY KEY,
31     first_name          VARCHAR(50),
32     last_name           VARCHAR(50),
33     DOB                 DATE,
34     gender              VARCHAR(10),
35     address             VARCHAR(100),
36     phone_no           VARCHAR(15),
37
38     CHECK (gender in ('Male', 'Female', 'Other','male', 'female', 'other'))
39 );
40

```

Figure 3

Source: screenshot from “postgresql Pgadmin4”

We have generated another table with a unique ID for the instructor which is created by a sequence named instructor ID sequence(inst_id_seq). and most of the values are similar to that

of member_details so in another way we can generate different tables for addresses and phone numbers to reduce the complexity of the table since we have few number of instructors this is not advisable for the database as may slow down as a whole database.

Figure 4

Source: screenshot from “postgresql Pgadmin4”

We now use realistic values to populate the instructor table using the function insert and for instructor ID we have an auto sequence which will generate a unique ID for the instructor.

```
43 CREATE TABLE plan_details
44 (
45     plan_code      VARCHAR(3) PRIMARY KEY,
46     plan_name      VARCHAR(50),
47     price          VARCHAR(10),
48     paln_duration  VARCHAR(10)
49 );
50
```

Figure 5

Source: screenshot from “postgresql Pgadmin4”

The above picture shows the construction of next table which is plan details which consists of a unique code for each plan and a plan name along with the price and the planed duration.

```
62
63 CREATE TABLE pay_n_renewal
64 (
65     payment_id      INT PRIMARY KEY,
66     member_id       TEXT,
67     plan_code       VARCHAR(3),
68     payment_type    VARCHAR(20),
69     payment_date    DATE,
70
71     CHECK (payment_type IN ('cash','online','card','cheque')),
72
73     FOREIGN KEY (member_id) REFERENCES member_details (member_id),
74     FOREIGN KEY (plan_code) REFERENCES plan_details (plan_code)
75
76 );
77
```

This above table is named as pay and renewal and this table consists of details regarding the payment which is made by a member along with the plan code date and the payment type. Here the member ID and the plan code are foreign keys and payment ID is a primary key. stop

```
79 create table enrollement
80 (
81
82     member_id TEXT REFERENCES member_details (member_id) ON UPDATE CASCADE ON DELETE CASCADE,
83     class_code varchar(10) REFERENCES class_details (class_code) ON UPDATE CASCADE,
84
85     PRIMARY KEY (member_id, class_code)          --composite primary key
86 );
87
88
```

This type of table is known as enrollment which stores the member ID and the class code which indicates that which member is taking which class and both of them combined generate a composite primary key which makes the data unique and we also have used a token named as cascade which allows us to change the member ID or class code when the parent table is updated or deleted this helps to accurately calculate number of members per class.

```
90
91 create table booking
92 (
93     member_id text,
94     class_code VARCHAR(10),
95     room_no int,
96     start_date DATE,
97     end_date DATE,
98     FOREIGN KEY (member_id) REFERENCES member_details (member_id),
99     FOREIGN KEY (class_code) REFERENCES class_details (class_code)
100 );
101
```

Figure 6

Source: screenshot from “postgresql Pgadmin4”

Above table consists data rooms booked by the member for a particular class. Since there are only a few plans available right now which are specified by the ALC we use that as a reference and populate the table.

```
52  
53 CREATE TABLE class_details  
54 (  
55     class_code      VARCHAR(10) PRIMARY KEY,  
56     class_name      VARCHAR(50),  
57     room_no         INT,  
58     instructor_id    text not null,  
59     FOREIGN KEY (instructor_id) REFERENCES instructor_details (instructor_id)  
60 );
```

Figure 7

Source: screenshot from “postgresql Pgadmin4”

This table contains the details of the class and each class has an unique code along with the class name room number instructor ID and here the instructor ID is used as a foreign key which references to the table instructor details

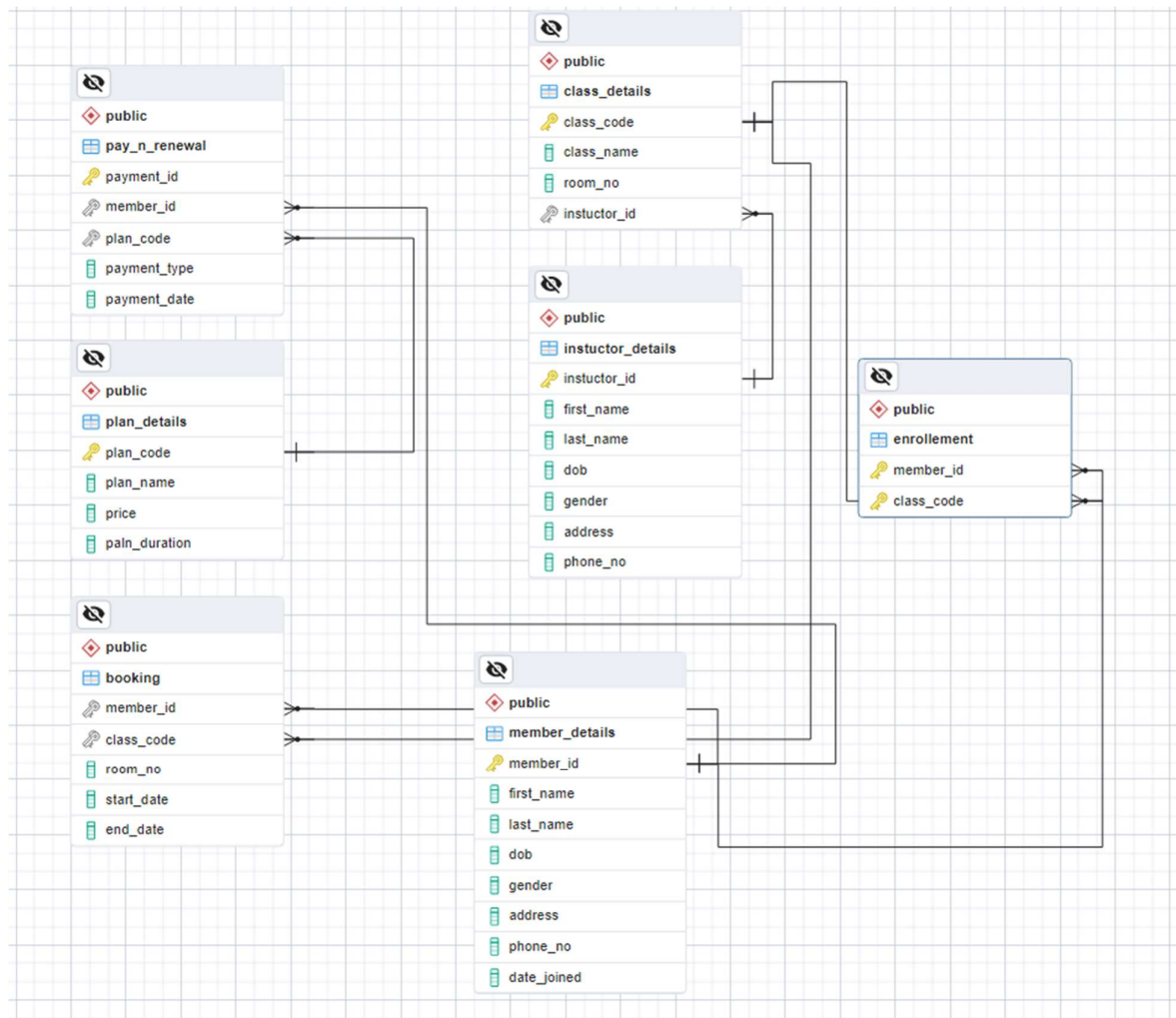
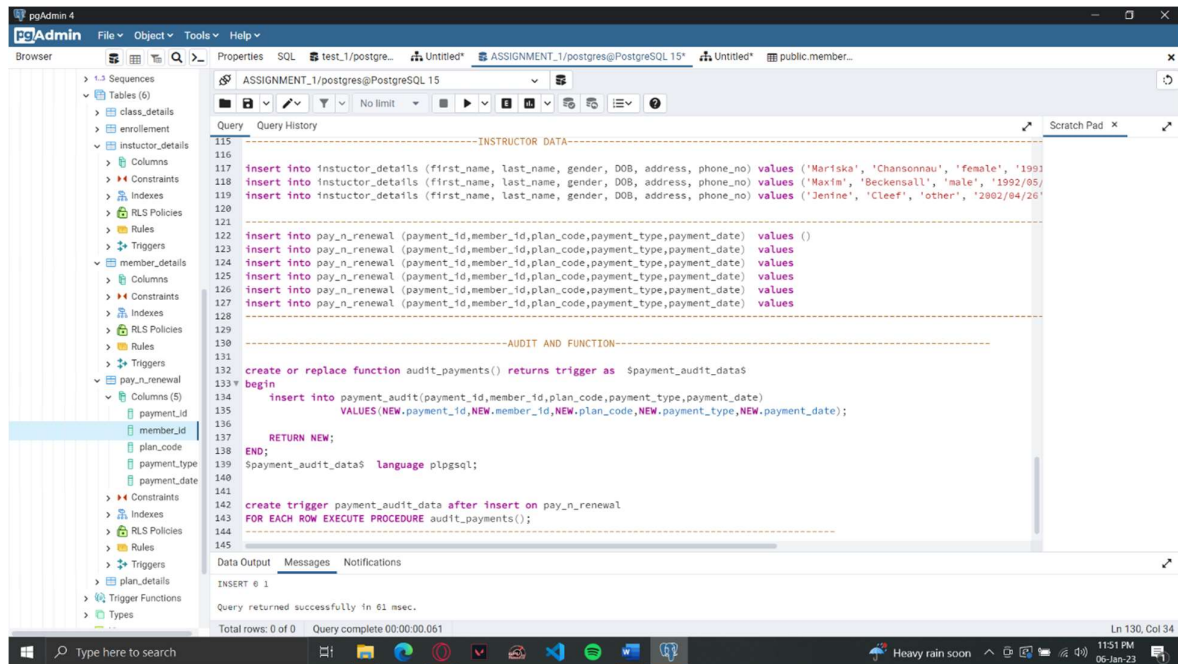


Figure 12: ERD diagram

Source: screenshot from “postgresql Pgadmin4”

The Entity Relationship diagram is generated using generate ERD tool which is available in postgresql pgadmin4.



References

Data generated by <https://www.mockaroo.com>