CPSC304 M4 deliverable

This PDF contains information including:
a. A short description of the final project, and what it accomplished.

Our project allows users to manage a gym business which involves managing members, trainers, the cafe inside the gyms, as well as the food sold at the cafes. The users are able to insert or delete rows to tables, as well as modify existing data or aggregate data for the purposes of finding meaningful data.

b. A description of how your final schema differed from the schema you turned in.
i. If the final schema differed, explain why. Note that turning in a final schema that's different from what you planned is fine, we just want to know what changed and why.

We changed some parts of the schema related to foreign keys and dropping or inserting values into tables. There was an issue of circular references happening with some of the tables, such as a gym and manager who each held a foreign key referencing the other. This created an issue for inserting or deleting the tables; for example, it would not allow inserting a new manager who worked at a particular gym that didn't exist yet and we also could not insert a new gym that was managed by that new manager. So we set the circular referencing foreign keys to be DEFERRABLE constraints and also set foreign keys to cascade on delete.

c. A copy of the schema and screenshots that show what data is present in each relation after the SQL script from item #2 is run.

Gym

```
postgres=# \d gym
                    Table "public.gym"
  Column   |     Type      | Collation | Nullable | Default
-----------+---------------+-----------+----------+---------
 branchnum | integer       |           | not null |
 capacity  | integer       |           |          |
 city      | character(20) |           |          |
 mid       | integer       |           |          |
Indexes:
    "gym_pkey" PRIMARY KEY, btree (branchnum)
Foreign-key constraints:
    "fk_mid" FOREIGN KEY (mid) REFERENCES manager(mid) DEFERRABLE
Referenced by:
    TABLE "cafe" CONSTRAINT "fk_branchnum" FOREIGN KEY (branchnum) REFERENCES gym(branchnum) DEFERRABLE
    TABLE "manager" CONSTRAINT "fk_gymnum" FOREIGN KEY (gymnum) REFERENCES gym(branchnum) DEFERRABLE
    TABLE "member" CONSTRAINT "member_branchnum_fkey" FOREIGN KEY (branchnum) REFERENCES gym(branchnum)
    TABLE "worksat" CONSTRAINT "worksat_branchnum_fkey" FOREIGN KEY (branchnum) REFERENCES gym(branchnum)
```

```
postgres=# select * from gym;
 branchnum | capacity |         city          |  mid
-----------+----------+-----------------------+-------
       111 |      150 | Vancouver             | 12345
       222 |      200 | Victoria              | 23456
       333 |      100 | Vancouver             | 34567
       444 |      120 | Surrey                | 45678
       555 |       50 | Vancouver             | 56789
       666 |       20 | Kamloops              | 67890
       777 |      110 | Vancouver             | 11111
       888 |      250 | Surrey                | 22222
       999 |      180 | Surrey                | 33333
       122 |       90 | Victoria              | 44444
       133 |       80 | Kamloops              | 55555
       144 |      105 | Victoria              | 66666
       155 |       55 | Surrey                | 77777
       166 |       10 | Kamloops              | 88888
(14 rows)
```

Manager

```
postgres=# \d manager;
                Table "public.manager"
 Column |     Type      | Collation | Nullable | Default
--------+---------------+-----------+----------+---------
 mid    | integer       |           | not null |
 mname  | character(20) |           |          |
 gymnum | integer       |           |          |
Indexes:
    "manager_pkey" PRIMARY KEY, btree (mid)
Foreign-key constraints:
    "fk_gymnum" FOREIGN KEY (gymnum) REFERENCES gym(branchnum) DEFERRABLE
Referenced by:
    TABLE "gym" CONSTRAINT "fk_mid" FOREIGN KEY (mid) REFERENCES manager(mid) DEFERRABLE
```

```
postgres=# select * from manager;
  mid  |        mname         | gymnum
-------+----------------------+--------
 12345 | Jon                  |    111
 23456 | Sara                 |    222
 34567 | Ash                  |    333
 45678 | Bill                 |    444
 56789 | Avery                |    555
 67890 | Cait                 |    666
 11111 | Kayn                 |    777
 22222 | Bruno                |    888
 33333 | Jack                 |    999
 44444 | Sam                  |    122
 55555 | Frank                |    133
 66666 | Juno                 |    144
 77777 | Cass                 |    155
 88888 | Karma                |    166
(14 rows)
```

Trainer

```
postgres=# \d trainer;
                Table "public.trainer"
 Column |     Type      | Collation | Nullable | Default
--------+---------------+-----------+----------+---------
 tid    | integer       |           | not null |
 tname  | character(20) |           |          |
Indexes:
    "trainer_pkey" PRIMARY KEY, btree (tid)
Referenced by:
    TABLE "trains" CONSTRAINT "trains_tid_fkey" FOREIGN KEY (tid) REFERENCES trainer(tid)
    TABLE "worksat" CONSTRAINT "worksat_tid_fkey" FOREIGN KEY (tid) REFERENCES trainer(tid)
```

```
postgres=# select * from trainer;
 tid |        tname
-----+---------------------
 123 | Sara
 393 | Grace
 321 | Warren
 251 | Gary
 920 | Trish
 295 | Bella
 314 | Yan
 934 | Yan
(8 rows)
```

Worksat

```
postgres=# \d worksat
                    Table "public.worksat"
     Column     |     Type      | Collation | Nullable | Default
----------------+---------------+-----------+----------+---------
 branchnum      | integer       |           | not null |
 tid            | integer       |           | not null |
 employmenttype | character(20) |           |          |
Indexes:
    "worksat_pkey" PRIMARY KEY, btree (branchnum, tid)
Foreign-key constraints:
    "worksat_branchnum_fkey" FOREIGN KEY (branchnum) REFERENCES gym(branchnum)
    "worksat_tid_fkey" FOREIGN KEY (tid) REFERENCES trainer(tid)
```

```
postgres=# select * from worksat;
 branchnum | tid |    employmenttype
-----------+-----+----------------------
       111 | 123 | Full-Time
       222 | 251 | Part-Time
       333 | 321 | Casual
       444 | 393 | Full-Time
       555 | 920 | Casual
       666 | 314 | Part-Time
       333 | 295 | Full-Time
       111 | 934 | Full-Time
(8 rows)
```

Member

```
postgres=# \d member;
                   Table "public.member"
    Column     |     Type      | Collation | Nullable | Default
---------------+---------------+-----------+----------+---------
 memid         | integer       |           | not null |
 phonenum      | bigint        |           |          |
 streetaddr    | character(20) |           |          |
 memname       | character(20) |           |          |
 membershipnum | integer       |           |          |
 branchnum     | integer       |           |          |
Indexes:
    "member_pkey" PRIMARY KEY, btree (memid)
Foreign-key constraints:
    "fk_membershipnum" FOREIGN KEY (membershipnum) REFERENCES membership(memnum) DEFERRABLE
    "member_branchnum_fkey" FOREIGN KEY (branchnum) REFERENCES gym(branchnum)
Referenced by:
    TABLE "buys" CONSTRAINT "buys_memid_fkey" FOREIGN KEY (memid) REFERENCES member(memid)
    TABLE "membership" CONSTRAINT "fk_memid" FOREIGN KEY (memid) REFERENCES member(memid) DEFERRABLE
    TABLE "trains" CONSTRAINT "trains_memid_fkey" FOREIGN KEY (memid) REFERENCES member(memid)
    TABLE "uses" CONSTRAINT "uses_memid_fkey" FOREIGN KEY (memid) REFERENCES member(memid)
```

```
postgres=# select * from member;
 memid |  phonenum  |      streetaddr      |       memname        | membershipnum | branchnum
-------+------------+----------------------+----------------------+---------------+-----------
    11 | 8881234567 | 123 Some Street      | Jane Doe             |          1000 |       555
    22 | 8881112222 | 223 Test Avenue      | John Dee             |          1001 |       444
    33 | 8881231212 | 990 Fake Street      | Phil Frank           |          1002 |       333
    44 | 8881998823 | 102 Random Boulevard | Sara Jones           |          1003 |       222
    55 | 8882222277 | 98 Hello World       | Meredith Grey        |          1004 |       111
    66 | 2842349877 | 99 Street Name       | Blicker Jones        |          1005 |       555
    77 | 9999999999 | 12 Hello Ave         | Dove Boot            |          1006 |       555
    88 | 2222333333 | 93 Fire Ave          | Mr Grinch            |          1007 |       666
    99 | 1200394129 | 94 Earth Ave         | Harry Potter         |          1008 |       777
    12 | 2912435234 | 95 Water Ave         | Ginny Weasley        |          1009 |       888
    13 | 1232131233 | 161 Air Ave          | Lucy Hale            |          1010 |       999
    14 | 9458202340 | 11 One Street        | Oprah Winfrey        |          1011 |       122
    15 | 2340325252 | 22 Two Street        | Rilakkuma            |          1012 |       133
    16 | 1306948593 | 33 Three Street      | Tiger Balm           |          1013 |       144
    17 | 4852020344 | 44 Four Street       | Phil Knight          |          1014 |       155
    18 | 4838371013 | 555 Five Street      | Santa Ono            |          1015 |       166
(16 rows)
```

Trains

```
postgres=# \d trains;
                Table "public.trains"
 Column |  Type   | Collation | Nullable | Default
--------+---------+-----------+----------+---------
 tid    | integer |           | not null |
 memid  | integer |           | not null |
Indexes:
    "trains_pkey" PRIMARY KEY, btree (tid, memid)
Foreign-key constraints:
    "trains_memid_fkey" FOREIGN KEY (memid) REFERENCES member(memid)
    "trains_tid_fkey" FOREIGN KEY (tid) REFERENCES trainer(tid)
```

```
postgres=# select * from trains;
 tid | memid
-----+-------
 123 |    11
 251 |    22
 321 |    33
 393 |    44
 920 |    55
 314 |    66
 314 |    11
 314 |    18
 123 |    44
 251 |    99
 251 |    55
 393 |    15
 920 |    33
 920 |    18
 251 |    11
 295 |    11
 321 |    11
 393 |    11
 920 |    11
 934 |    11
(20 rows)
```

Membership

```
postgres=# \d membership
                  Table "public.membership"
     Column      |     Type      | Collation | Nullable | Default
-----------------+---------------+-----------+----------+---------
 memnum          | integer       |           | not null |
 expirydate      | character(20) |           |          |
 memid           | integer       |           |          |
 amenityaccess   | boolean       |           |          |
 hasmassage      | boolean       |           |          |
 hasfitnessclass | boolean       |           |          |
Indexes:
    "membership_pkey" PRIMARY KEY, btree (memnum)
Foreign-key constraints:
    "fk_memid" FOREIGN KEY (memid) REFERENCES member(memid) DEFERRABLE
Referenced by:
    TABLE "member" CONSTRAINT "fk_membershipnum" FOREIGN KEY (membershipnum) REFERENCES membership(memnum) DEFERRABLE
```

```
postgres=# select * from membership;
 memnum |      expirydate      | memid | amenityaccess | hasmassage | hasfitnessclass
--------+----------------------+-------+---------------+------------+-----------------
   1000 | 12/02/2022           |    11 | t             | t          | t
   1001 | 12/12/2023           |    22 | f             | f          | t
   1002 | 02/01/2023           |    33 | f             | f          | f
   1003 | 01/01/2024           |    44 | t             | f          | t
   1004 | 05/12/2023           |    55 | f             | t          | t
   1005 | 01/01/2025           |    66 | t             | t          | t
   1006 | 10/10/2023           |    77 | f             | f          | f
   1007 | 11/11/2022           |    88 | f             | f          | f
   1008 | 11/11/2022           |    99 | f             | f          | f
   1009 | 10/10/2024           |    12 | t             | f          | f
   1010 | 02/02/2023           |    13 | t             | t          | t
   1011 | 03/03/2022           |    14 | f             | t          | t
   1012 | 04/04/2024           |    15 | f             | t          | t
   1013 | 05/05/2025           |    16 | t             | f          | t
   1014 | 06/06/2026           |    17 | f             | t          | f
   1015 | 07/07/2027           |    18 | t             | f          | f
(16 rows)
```

Equipment

```
postgres=# \d equipment
                Table "public.equipment"
  Column   |     Type      | Collation | Nullable | Default
-----------+---------------+-----------+----------+---------
 serialnum | integer       |           | not null |
 ename     | character(20) |           |          |
 etype     | character(20) |           |          |
 estatus   | character(20) |           |          |
Indexes:
    "equipment_pkey" PRIMARY KEY, btree (serialnum)
Referenced by:
    TABLE "uses" CONSTRAINT "uses_serialnum_fkey" FOREIGN KEY (serialnum) REFERENCES equipment(serialnum)
```

```
postgres=# select * from equipment;
 serialnum |        ename         |        etype         |       estatus
-----------+----------------------+----------------------+----------------------
     12345 | Dumbbell             | Weight               | Available
     13131 | Kettlebell           | Weight               | Enroute
     23232 | Treadmill            | Cardio               | Available
     45677 | Stairmaster          | Cardio               | Available
     23432 | Rowing Machine       | Cardio               | Repair
     92882 | Bench                | Strength             | Enroute
     19201 | Barbell              | Strength             | Available
(7 rows)
```

Uses

```
postgres=# \d uses
                  Table "public.uses"
  Column   |     Type      | Collation | Nullable | Default
-----------+---------------+-----------+----------+---------
 routine   | character(20) |           |          |
 serialnum | integer       |           | not null |
 memid     | integer       |           | not null |
Indexes:
    "uses_pkey" PRIMARY KEY, btree (serialnum, memid)
Foreign-key constraints:
    "uses_memid_fkey" FOREIGN KEY (memid) REFERENCES member(memid)
    "uses_serialnum_fkey" FOREIGN KEY (serialnum) REFERENCES equipment(serialnum)
```

```
postgres=# select * from uses;
       routine        | serialnum | memid
----------------------+-----------+-------
 Endurance            |     23232 |    11
 Strength             |     12345 |    22
 Burst                |     13131 |    33
 Rowing               |     23432 |    44
 Bodybuilding         |     45677 |    55
 Strength             |     19201 |    66
(6 rows)
```

Cafe

```
postgres=# \d cafe;
                Table "public.cafe"
  Column   |  Type   | Collation | Nullable | Default
-----------+---------+-----------+----------+---------
 storenum  | integer |           | not null |
 branchnum | integer |           | not null |
Indexes:
    "cafe_pkey" PRIMARY KEY, btree (storenum, branchnum)
Foreign-key constraints:
    "fk_branchnum" FOREIGN KEY (branchnum) REFERENCES gym(branchnum) DEFERRABLE
Referenced by:
    TABLE "food" CONSTRAINT "food_storenum_branchnum_fkey" FOREIGN KEY (storenum, branchnum) REFERENCES cafe(storenum, branchnum)
```

```
postgres=# select * from cafe;
 storenum | branchnum
----------+-----------
   329857 |       111
   398470 |       222
   122143 |       333
   519281 |       444
   273463 |       555
   192931 |       666
(6 rows)
```

Food

```
postgres=# \d food;
                    Table "public.food"
  Column   |       Type       | Collation | Nullable | Default
-----------+------------------+-----------+----------+---------
 fid       | integer          |           | not null |
 price     | double precision |           |          |
 storenum  | integer          |           |          |
 branchnum | integer          |           |          |
Indexes:
    "food_pkey" PRIMARY KEY, btree (fid)
Foreign-key constraints:
    "food_storenum_branchnum_fkey" FOREIGN KEY (storenum, branchnum) REFERENCES cafe(storenum, branchnum)
Referenced by:
    TABLE "buys" CONSTRAINT "buys_fid_fkey" FOREIGN KEY (fid) REFERENCES food(fid) ON DELETE CASCADE
```

```
postgres=# select * from food;
 fid  | price  | storenum | branchnum
------+--------+----------+-----------
 3434 |  12.99 |   329857 |       111
 3997 |     10 |   398470 |       222
 3883 |   8.98 |   122143 |       333
 2412 |   5.59 |   519281 |       444
 1269 |  14.49 |   273463 |       555
 1092 |  25.99 |   192931 |       666
 1882 |   3.99 |   192931 |       666
 1094 |  12.92 |   519281 |       444
 1902 |  20.45 |   273463 |       555
 5991 |  29.99 |   122143 |       333
 6931 |   6.69 |   192931 |       666
 9041 |  59.95 |   329857 |       111
 9402 |  31.31 |   398470 |       222
 5810 |    8.9 |   273463 |       555
 8120 |   3.99 |   329857 |       111
 8281 | 129.99 |   329857 |       111
 2345 |  45.42 |   398470 |       222
 6211 |      7 |   122143 |       333
 1966 |  15.59 |   519281 |       444
 6010 |    9.1 |   192931 |       666
(20 rows)
```

Buys

```
postgres=# \d buys
                  Table "public.buys"
 Column |  Type   | Collation | Nullable | Default
--------+---------+-----------+----------+---------
 memid  | integer |           | not null |
 fid    | integer |           | not null |
Indexes:
    "buys_pkey" PRIMARY KEY, btree (memid, fid)
Foreign-key constraints:
    "buys_fid_fkey" FOREIGN KEY (fid) REFERENCES food(fid) ON DELETE CASCADE
    "buys_memid_fkey" FOREIGN KEY (memid) REFERENCES member(memid)
```

```
postgres=# select * from buys;
 memid | fid
-------+------
    11 | 1269
    22 | 2412
    33 | 3434
    44 | 3883
    55 | 3997
    66 | 1092
(6 rows)
```

d. A list of all SQL queries used. For SQL query requirements, check the rubric listed on Canvas for Milestone 4.

INSERT

```
// INSERT QUERY
// create food
app.post("/food", async(req, res) => {
    try {
        const { fid, price, storenum, branchnum } = req.body;
        const newFood = await pool.query(
            "INSERT INTO food (fid, price, storenum, branchnum) VALUES($1, $2, $3, $4) RETURNING *", [fid, price, storenum, branchnum]);
        res.json(newFood.rows[0]);
    } catch (err) {
        console.log(err.message);
    }
});
```

DELETE

```
// DELETE QUERY
// delete food
app.delete("/food/:id", async(req, res) =>{
    try {
        const { id } = req.params;
        const deleteFood = await pool.query(
            "DELETE FROM food WHERE fid = $1", [id]);
        res.json("Food was deleted");
    } catch (error) {
        console.log(err.message);
    }
})
```

UPDATE

```
// UPDATE QUERY
// update food price
app.put("/food/:id", async (req, res) => {
    try {
        const { id } = req.params;
        const { price } = req.body;
        const updateFood = await pool.query(
            "UPDATE food SET price = $1 WHERE fid = $2",
            [price, id]);
        res.json("Food price updated!")
    } catch (err) {
        console.log(err.message);
    }
});
```

Selection

```
// SELECTION QUERY
// all members who go to a particular branch and membership ID > x
app.get("/member/:branchnum/:memid", async (req, res) => {
    try {
        const { branchnum, memid } = req.params;
        const members = await pool.query(
            "SELECT * FROM member WHERE branchnum = $1 AND memid > $2", [branchnum, memid]);
        res.json(members.rows);
    } catch (err) {
        console.log(err.message);
    }
});
```

Projection

```
// PROJECTION QUERY
// get memnum, memid, expirydate of existing memberships
app.get("/membership", async (req, res) => {
    try {
        const membership = await pool.query(
            "SELECT memnum, memid, expirydate FROM membership");
        res.json(membership.rows);
    } catch (err) {
        console.log(err.message);
    }
});
```

Join

```
// JOIN QUERY
// get trainer id, member id, and equipment serial number given a specific member routine
app.get("/uses/:routine", async (req, res) => {
    try {
        const { routine } = req.params;
        const routineInfo = await pool.query(
            "SELECT t.tid, u.memid, u.serialnum FROM trains t, uses u WHERE u.routine = $1 AND t.memid = u.memid", [routine]);
        res.json(routineInfo.rows);
    } catch (err) {
        console.log(err.message);
    }
});
```

Aggregation with GROUP BY

```
// AGGREGATE WITH GROUP BY QUERY
// get cafe store number and sum of food prices grouped by storenum
app.get("/foodsum", async (req, res) => {
    try {
        const food = await pool.query(
            "SELECT storenum, sum(price) FROM food GROUP BY storenum");
        res.json(food.rows);
    } catch (err) {
        console.log(err.message);
    }
});
```

Aggregation with HAVING

```
// AGGREGATE WITH HAVING QUERY
// get most expensive item from each store that sells more than 1 item
app.get("/fooditem", async (req, res) => {
    try {
        const food = await pool.query(
            "SELECT max(price), storenum FROM food GROUP BY storenum HAVING count(price) > 1");
        res.json(food.rows);
    } catch (err) {
        console.log(err.message);
    }
});
```

Nested aggregation with GROUP BY

```
create view temp(city, cap) as select g.city, avg(g.capacity) as cap from gym g group by g.city;
```

```
// NESTED AGGREGATION WITH GROUP BY QUERY
// get city and average gym capacity for which the average is the minimum over all gyms by city
app.get("/mingym", async (req, res) => {
    try {
        const gym = await pool.query(
            "select city, cap from temp where cap = (select min(cap) from temp)");
        res.json(gym.rows);
    } catch (err) {
        console.log(err.message);
    }
});
```

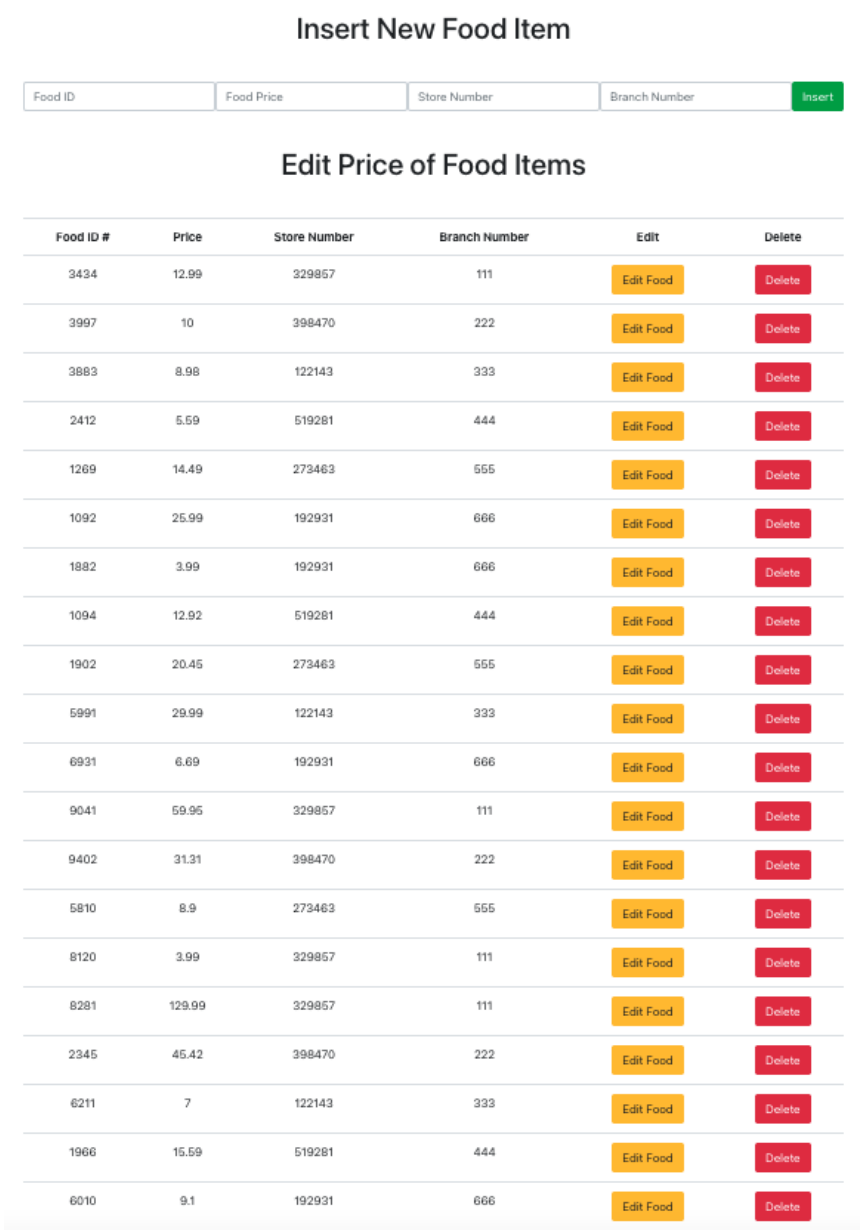
Division

```
// DIVISION QUERY
// get members trained by all trainers
app.get("/trainsall", async (req, res) => {
    try {
        const trainall = await pool.query(
            "select m.memname, m.memid from member m where not exists (select tr.tid from trainer tr except (select tr2.tid from trains tr2 where tr2.memid = m.memid))");
        res.json(trainall.rows);
    } catch (err) {
        console.log(err.message);
    }
});
```

e. Screenshots of the sample output of the queries using the GUI (for example, you can show what data is in your table before you run the query, and then show another screenshot after running the query, from some kind of GUI input like a button).

i. You need only to include screenshots for the required queries – if you implemented more than what was required, screenshots are not needed for those extra queries.

INSERT:



Enter the values to insert. In our example, we have inserted 1, 2, 329857, 111

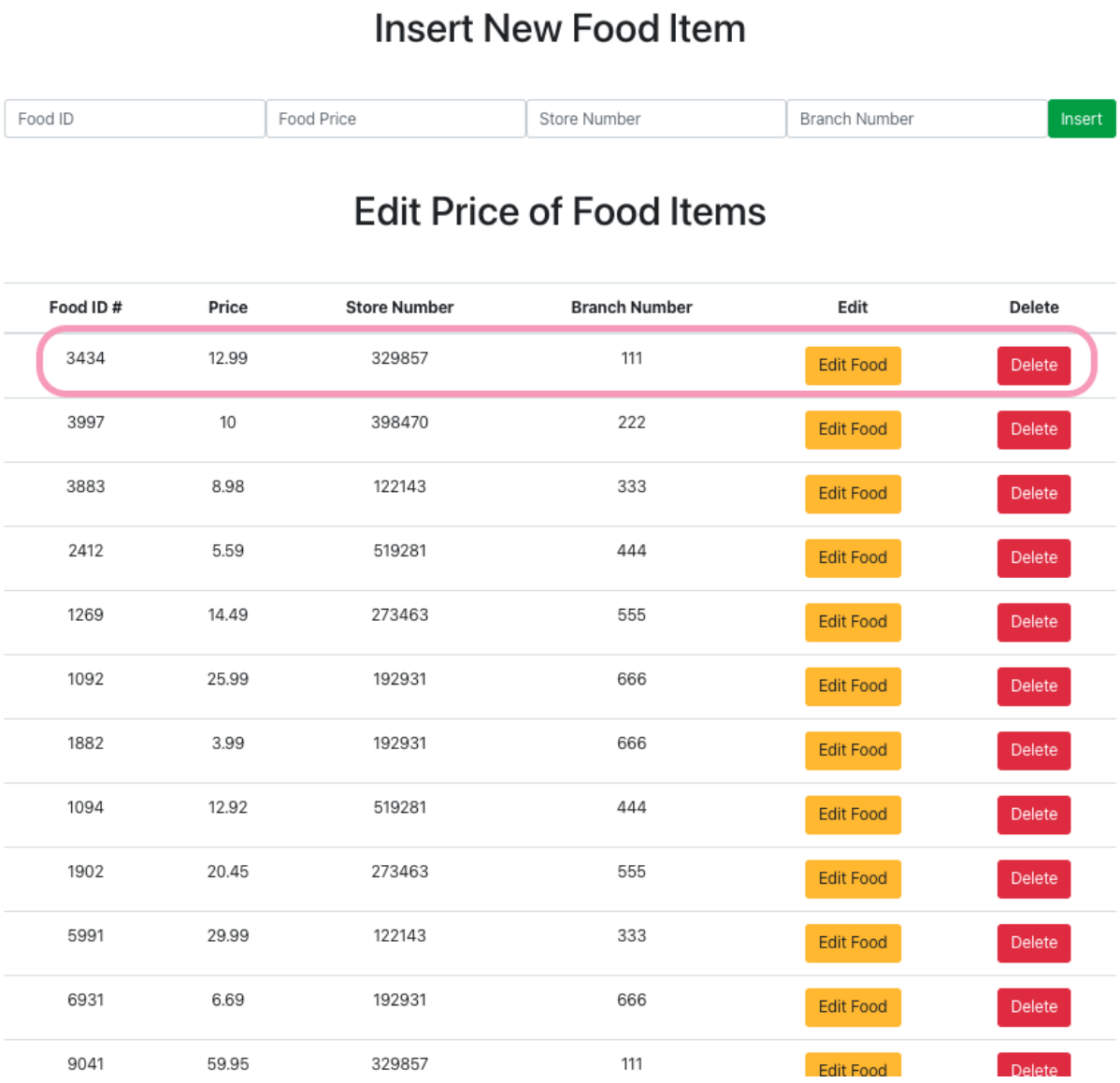
After inputting the values into their respective text boxes and clicking “Insert”, the new data is shown in the table. The new input is highlighted at the bottom.

Edit Price of Food Items

Food ID #	Price	Store Number	Branch Number	Edit	Delete
3434	12.99	329857	111	Edit Food	Delete
3997	10	398470	222	Edit Food	Delete
3883	8.98	122143	333	Edit Food	Delete
2412	5.59	519281	444	Edit Food	Delete
1269	14.49	273463	555	Edit Food	Delete
1092	25.99	192931	666	Edit Food	Delete
1882	3.99	192931	666	Edit Food	Delete
1094	12.92	519281	444	Edit Food	Delete
1902	20.45	273463	555	Edit Food	Delete
5991	29.99	122143	333	Edit Food	Delete
6931	6.69	192931	666	Edit Food	Delete
9041	59.95	329857	111	Edit Food	Delete
9402	31.31	398470	222	Edit Food	Delete
5810	8.9	273463	555	Edit Food	Delete
8120	3.99	329857	111	Edit Food	Delete
8281	129.99	329857	111	Edit Food	Delete
2345	45.42	398470	222	Edit Food	Delete
6211	7	122143	333	Edit Food	Delete
1966	15.59	519281	444	Edit Food	Delete
6010	9.1	192931	666	Edit Food	Delete
1	2	329857	111	Edit Food	Delete

DELETE:

The state of the application is what is shown in the screenshot at the end of Insert. If we wanted to delete a food item (Food ID #3434 for example), we simply click the Delete button in the row with Food ID #3434.



Now the table looks like this:

Insert New Food Item

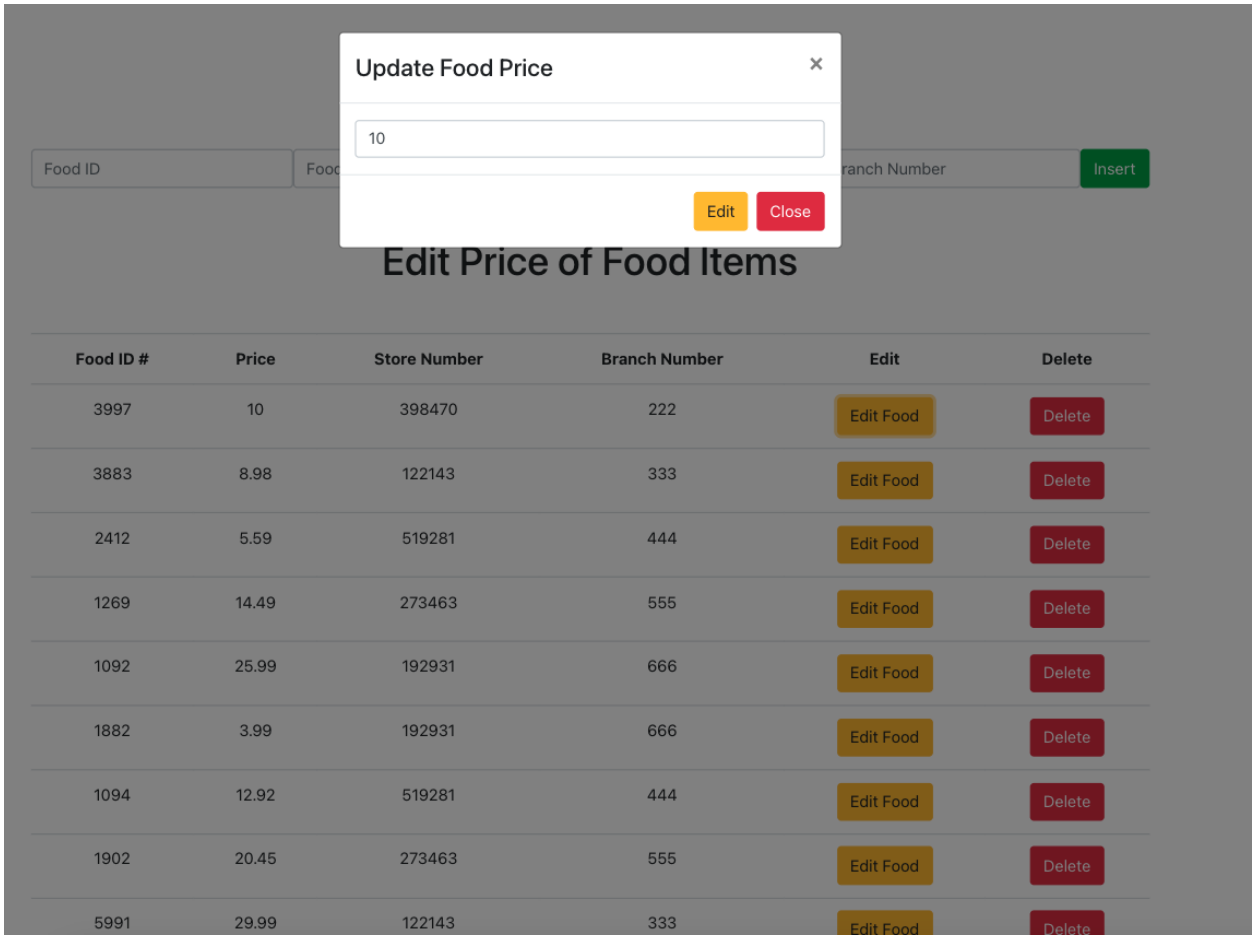
Food ID | Food Price | Store Number | Branch Number | Insert

Edit Price of Food Items

Food ID #	Price	Store Number	Branch Number	Edit	Delete
3997	10	398470	222	Edit Food	Delete
3883	8.98	122143	333	Edit Food	Delete
2412	5.59	519281	444	Edit Food	Delete
1269	14.49	273463	555	Edit Food	Delete
1092	25.99	192931	666	Edit Food	Delete
1882	3.99	192931	666	Edit Food	Delete
1094	12.92	519281	444	Edit Food	Delete
1902	20.45	273463	555	Edit Food	Delete
5991	29.99	122143	333	Edit Food	Delete
6931	6.69	192931	666	Edit Food	Delete
9041	59.95	329857	111	Edit Food	Delete
9402	31.31	398470	222	Edit Food	Delete
5810	8.9	273463	555	Edit Food	Delete
8120	3.99	329857	111	Edit Food	Delete
8281	129.99	329857	111	Edit Food	Delete
2345	45.42	398470	222	Edit Food	Delete
6211	7	122143	333	Edit Food	Delete
1966	15.59	519281	444	Edit Food	Delete
6010	9.1	192931	666	Edit Food	Delete
1	2	329857	111	Edit Food	Delete

The row with Food ID #3434 is now deleted from the database.

UPDATE:
The state of the application is what is shown in the image at the end of Delete. Now we want to update the price of a food item. If we wanted to update a food item with ID #3997 to have a price of 11 instead of 10, we can simply click the ‘Edit Food’ button in the same row as Food ID #3997. A popup will appear which will allow you to change the price of the food item.



The box will show the current price of the food but you can input the new price into the text box, click edit, the price will be updated.

Edit Price of Food Items

Food ID #	Price	Store Number	Branch Number	Edit	Delete
3883	8.98	122143	333	Edit Food	Delete
2412	5.59	519281	444	Edit Food	Delete
1269	14.49	273463	555	Edit Food	Delete
1092	25.99	192931	666	Edit Food	Delete
1882	3.99	192931	666	Edit Food	Delete
1094	12.92	519281	444	Edit Food	Delete
1902	20.45	273463	555	Edit Food	Delete
5991	29.99	122143	333	Edit Food	Delete
6931	6.69	192931	666	Edit Food	Delete
9041	59.95	329857	111	Edit Food	Delete
9402	31.31	398470	222	Edit Food	Delete
5810	8.9	273463	555	Edit Food	Delete
8120	3.99	329857	111	Edit Food	Delete
8281	129.99	329857	111	Edit Food	Delete
2345	45.42	398470	222	Edit Food	Delete
6211	7	122143	333	Edit Food	Delete
1966	15.59	519281	444	Edit Food	Delete
6010	9.1	192931	666	Edit Food	Delete
1	2	329857	111	Edit Food	Delete
3997	11	398470	222	Edit Food	Delete

Now, the Food ID #3997 can be seen to have a price of 11, which is what we changed it to.

Aggregate with GROUP BY:
Our aggregate with group by works by getting the sum of all the prices of food in that specific store (having the same Store Number). Branch number was not referenced as both `storenum` and `branchnum` reference the entity cafe.

It can be seen that the Store Number is grouped and the sum of the prices in the store is shown.

Sum of all food prices grouped by Store Number

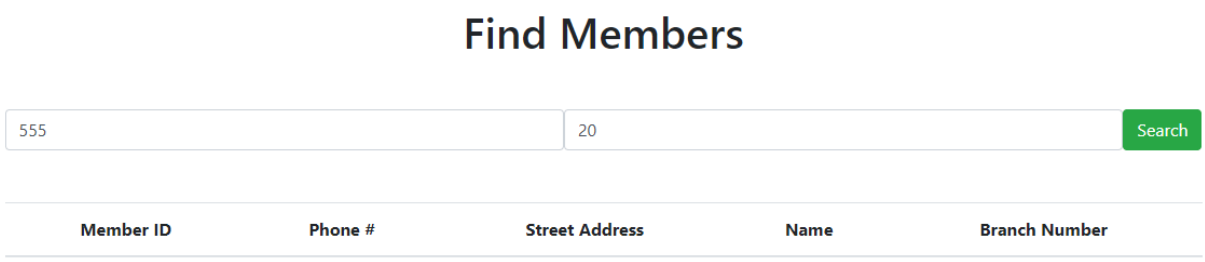
Store Number	Sum of Price
519281	34.10
398470	87.73
122143	45.97
192931	45.77
273463	43.84
329857	195.93

We can add another item (via the insert method from above) and see that the results will update the sum food price table. If we add food items with values (2, 100, 122143, 333), (3, 50, 398470, 222), we should see that Store Number 122143's sum will increase by 100 to be 145.97 and Store Number 398470's sum will increase by 50 to be 137.73.

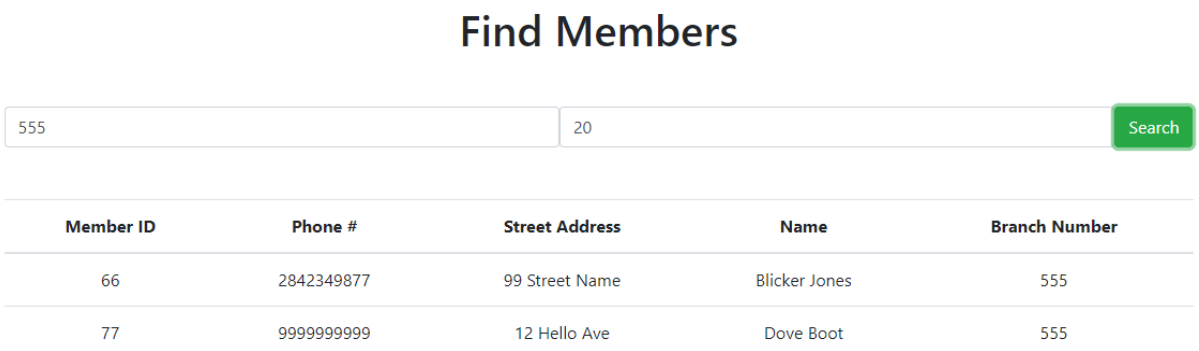
Sum of all food prices grouped by Store Number

Store Number	Sum of Price
519281	34.10
398470	137.73
122143	145.97
192931	45.77
273463	43.84
329857	195.93

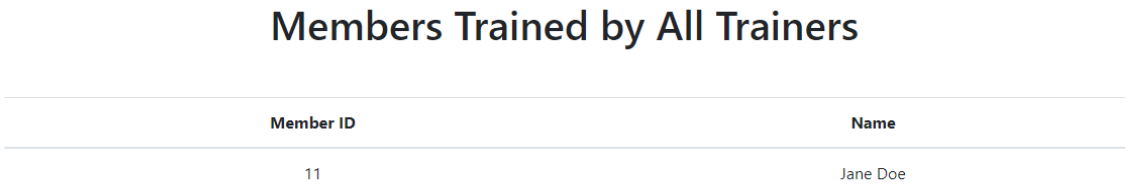
Select:
There is a box for searching for all members who have a specific branch number and a member ID > a certain amount. Below is a screenshot showing what this looks like before we have done a search…



If we do an example search on branch number 555 and ID 20, we get all members selected with a branch number = 555 and ID > 20 as shown:



Division:
Our division finds all members who are being trained by every trainer. In our sample data, the only example of this is Jane Doe:



Projection:
Our projection allows you to view the member ID, membership #, and expiry dates of all memberships currently stored in the database:

Membership Expiry Dates

Member ID	Membership Number	Expiry Date
11	1000	12/02/2022
22	1001	12/12/2023
33	1002	02/01/2023
44	1003	01/01/2024
55	1004	05/12/2023
66	1005	01/01/2025
77	1006	10/10/2023
88	1007	11/11/2022
99	1008	11/11/2022
12	1009	10/10/2024
13	1010	02/02/2023
14	1011	03/03/2022
15	1012	04/04/2024
16	1013	05/05/2025
17	1014	06/06/2026
18	1015	07/07/2027

Join:

The join query in our project allows you to find trainers and members who utilize a certain routine by specifying the type of routine in the search box. Below are a few examples of different types of routines being used in our data and their results:

Find Routines

Strength Search

Trainer ID	Member ID	Equipment Serial Number
251	22	12345
314	66	19201

Find Routines

Bodybuilding | Search

Trainer ID	Member ID	Equipment Serial Number
920	55	45677
251	55	45677

Aggregation with Having:

Our project uses aggregation with having to find the most expensive food item for each store and then display them in a table:

Most Expensive Food Item Per Store

Store Number	Item Price
519281	15.59
398470	45.42
122143	29.99
192931	25.99
273463	20.45
329857	129.99

If I were to insert a new food item with price 20 for the first store, the table updates:

Most Expensive Food Item Per Store

Store Number	Item Price
519281	20
398470	45.42
122143	29.99
192931	25.99
273463	20.45
329857	129.99

Nested Aggregation with GROUP BY:
The nested aggregation with GROUP BY first finds the average capacity of each gym for each city. It then returns the city with the lowest average as well as what that value is. For our test data, it was Kamloops:

Minimum Gym Capacity in a City

City	Minimum Average Capacity
Kamloops	36.6666666666666667