Delimiter Fitness Club

Joshua Budd

5/1/2016

My database has 7 entities included within it. The Members table is the first, with a primary key of ID to uniquely represent an individual member of the fitness club with the table used to store information about the member such as contact information and payment information. The second table is an associative table named item\_line, with a two keys that serve as the primary and foreign keys (MEMBERS$id and VENDORS$vname). Next is the vendors table, which has a primary key of vname and serves the purpose of keeping track of information about the different vendors that the club uses to purchase objects such as food and machinery. Following is the equipment table with a primary key of make\_model and a foreign key of VENDORS$vname, which relates to the venders table to store information about the different equipment that is used throughout the club and its condition. The next table is classes, which stores data about the different fitness classes offered at the club. It has a primary key of number, with 2 other primary and foreign keys of EMPLOYEES$ename and MEMBERS$id. Up next is the employees table that stores information such as position and contact information about the different employees at the fitness club. It has a primary key of ename. The last table is the personal\_training table. This table keeps tract of which members are enrolled in personal training sessions with the employee’s. It included fields that cover things such as recommended supplements and member weight. It has a 2 primary keys that are also foreign keys, EMPLOYEES$ename and MEMBERS$id.

Delimiter fitness club is a business that provides fitness advice, services, and equipment to its members in a convenient and logical way. The business works by having members sign up for a membership that gives them access to the different club facilities and services. The price of membership and services depend on the type of membership they sign up for. After they have a membership, the monthly fee and all other transactions are preformed through the use of their club account. A credit card is directly linked to their account, so funds can be instantly transferred from their credit card to their account. Members are free to use the club’s equipment, but are required to sign up for classes and personal training if they so desire. Employees are assigned roles when they are hired, and information about them (such as address and emergency contact information is stored) collected. Their responsibilities depend on what job they are assigned. Maintenance is required to repair and maintain the different equipment and ensure a clean environment is upheld. Trainers are responsible for the different classes offered through the club and the personal training sessions that they are assigned. The manager keeps tract of the other employee’s and makes sure that they are doing their job to the specifications required by the company policy and to ensure the continued operator of the club.

The data for the database for the Delimiter Fitness Club is entered by employees of the club. Such information is delegated to the employee responsible for that activity. The manager will enter employee information as well as the vendor information. Maintenance information and data is entered by the employee that is assigned to the “maintenance” job. For the member’s information, either the manager or the trainers may enter the information. All other information is generally entered into the system by the trainers or automatically through the vending machines that are connected to the club’s network. If a member wants to use their account to purchase something, they input the last 4 digits of the credit card that they used to sign up with to verify their identity and purchase something through the system. Most other information is entered through terminals located throughout the club that can only be accessed by employees.

During the course of this assignment, I did run into a few issues. One issue that I had was that I had too many entities in my ERD. I fixed this by moving some of the data in some entities around to limit the amount of tables that my table had. Another issue that I ran into was that I didn’t know what queries to make for my database. I asked myself “what information might need to be obtained from the information in the database”, which helped greatly in creating my queries.

Query 1: This query calculates what members of the club are enrolled in more than one class by searching the MEMBERS and the CLASSES tables and display the members ID, first name, and last name. The SQL code is:

SELECT MEMBERS.ID, MEMBERS.fname, MEMBERS.lname

FROM MEMBERS INNER JOIN CLASSES ON MEMBERS.ID = CLASSES.[MEMBERS$id]

GROUP BY MEMBERS.ID, MEMBERS.fname, MEMBERS.lname

HAVING (((Count(CLASSES.[MEMBERS$id]))>1));

| **Query1** | | |
| --- | --- | --- |
| **Members ID** | **fist name** | **last name** |
| 1009 | Tom | Flannigan |

The results are as follows:

The expected results are to display the members that are enrolled in more than one class.

Query 2: This query will search the MEMBERS table for applicable members that have the family type of membership and display their id, first name, last name, and membership type. The SQL code is:

SELECT MEMBERS.ID, MEMBERS.fname, MEMBERS.lname, MEMBERS.membership\_type

FROM MEMBERS

WHERE (((MEMBERS.membership\_type) Like "family"));

| **Query2** | | | |
| --- | --- | --- | --- |
| **Members ID** | **First name** | **Last name** | **Membership Type** |
| 1009 | Tom | Flannigan | family |
| 1010 | Siri | Flannigan | family |

The results are as follows:

The expected results are to display all members that have the “family” type of membership.

Query 3: This query will search the PERSONAL\_TRAINING table for how many sessions an employee is currently assigned to complete with members of the club and display the employees name and the number of sessions that they are currently apart of. The SQL code is as follows:

SELECT PERSONAL\_TRAINING.[EMPLOYEES$ename], Count(PERSONAL\_TRAINING.[EMPLOYEES$ename]) AS [CountOfEMPLOYEES$ename]

FROM MEMBERS INNER JOIN (EMPLOYEES INNER JOIN PERSONAL\_TRAINING ON EMPLOYEES.ename = PERSONAL\_TRAINING.[EMPLOYEES$ename]) ON MEMBERS.ID = PERSONAL\_TRAINING.[MEMBERS$id]

GROUP BY PERSONAL\_TRAINING.[EMPLOYEES$ename]

HAVING (((Count(PERSONAL\_TRAINING.[EMPLOYEES$ename]))>1));

| **Query3** | |
| --- | --- |
| **Employee's Name** | **Enrolled Sessions** |
| Gia Meckler | 2 |
| John Etton | 3 |

The results are as follows:

The expected results are that the name of the employee and their total number of enrolled sessions would be displayed.

Query 4: The purpose of this query is to search the EMPLOYEES table to find out which employees are in the position of “trainer” and make under $11.00 dollars an hour, then display the employees name, position, and rate. The SQL code is as follows:

SELECT EMPLOYEES.ename, EMPLOYEES.position, EMPLOYEES.rate

FROM EMPLOYEES

WHERE (((EMPLOYEES.position) Like "trainer") AND ((EMPLOYEES.rate)<11));

| **Query4** | | |
| --- | --- | --- |
| **Employee name** | **position** | **rate** |
| John Etton | trainer | $10.75 |

The results are as follows:

The expected results were the results of a search of all the employees who are trainers and who made less than $11.00 dollars an hour.

Query 5: This query will search the MEMBERS, VENDORS, and LINE\_ITEM tables to calculate the total amount of money that a particular member has spent on club services and display the member’s id, first name, last name, and the total amount that they have spent. The following is the SQL code:

SELECT DISTINCT ITEM\_LINE.[MEMBERS$id], MEMBERS.fname, MEMBERS.lname, Sum(ITEM\_LINE.sale) AS SumOfsale

FROM MEMBERS INNER JOIN (VENDORS INNER JOIN ITEM\_LINE ON VENDORS.vname = ITEM\_LINE.[VENDORS$vname]) ON MEMBERS.ID = ITEM\_LINE.[MEMBERS$id]

GROUP BY ITEM\_LINE.[MEMBERS$id], MEMBERS.fname, MEMBERS.lname

ORDER BY Sum(ITEM\_LINE.sale) DESC;

| **Query5** | | | |
| --- | --- | --- | --- |
| **Members ID** | **first name** | **last name** | **Total Spent** |
| 1011 | Piere | Hucklebottom | $19.80 |
| 1010 | Siri | Flannigan | $19.45 |
| 1007 | Mike | Touner | $11.90 |
| 1009 | Tom | Flannigan | $1.60 |

The results are as follows:

The expected results were to display the member’s information as well as the total amount that they have spent on club services.

Query 6: The purpose of this query is to calculate the amount of body fat that a member enrolled in personal training needs to lose to reach their goal body-to-fat ratio by pulling the information from the PERSONAL\_TRAINING tables and the MEMBERS table and display the member’s ID, first name, last name, and fat loss requirement. The SQL code is as follows:

SELECT DISTINCT PERSONAL\_TRAINING.[MEMBERS$id], MEMBERS.fname, MEMBERS.lname, [body\_to\_fat\_start]-[body\_to\_fat\_goal] AS [Fat Loss Requirement]

| **Query6** | | | |
| --- | --- | --- | --- |
| **Members ID** | **First name** | **Last name** | **Fat Loss Requirement** |
| 1008 | Leya | Scorth | 5.00% |
| 1009 | Tom | Flannigan | 3.00% |
| 1010 | Siri | Flannigan | 2.00% |
| 1011 | Piere | Hucklebottom | 3.00% |

FROM MEMBERS INNER JOIN PERSONAL\_TRAINING ON MEMBERS.ID = PERSONAL\_TRAINING.[MEMBERS$id];

The results are as follows:

The expected results were to display the amount of body fat that a member needed to lose in order to reach their body-to-fat goal.

Query 7: This query uses data from the MEMBERS table to display the member’s ID, first name, Last name, account balance, and payment type. The SQL code is as follows:

SELECT MEMBERS.ID, MEMBERS.fname, MEMBERS.lname, MEMBERS.account\_balance, MEMBERS.payment

FROM MEMBERS;

| **Query7** | | | | |
| --- | --- | --- | --- | --- |
| **ID** | **fname** | **lname** | **Account Balance** | **payment** |
| 1007 | Mike | Touner | $25.60 | master XXXXXXXXXXXX5643 |
| 1008 | Leya | Scorth | $0.00 | master XXXXXXXXXXXX6658 |
| 1009 | Tom | Flannigan | $500.00 | visa XXXXXXXXXXXX5647 |
| 1010 | Siri | Flannigan | $32.40 | master XXXXXXXXXXXX5643 |
| 1011 | Piere | Hucklebottom | $64.25 | master XXXXXXXXXXXX5641 |

The results are as follows:

The result is a list of all the members of the XYZ fitness club with their information and payment methods with their account balance.

Query 8: The following query searches the EQUIPMENT and VENDORS tables for which pieces of equipment needs maintenance and displays the equipment’s make/model, vendors name, the last time the equipment was maintained, and if it needs maintenance. The SQL is as follows:

SELECT EQUIPMENT.make\_model, EQUIPMENT.[VENDORS$vname], EQUIPMENT.last\_maintenance, EQUIPMENT.needed\_maintenance

FROM VENDORS INNER JOIN EQUIPMENT ON VENDORS.vname = EQUIPMENT.[VENDORS$vname]

WHERE (((EQUIPMENT.needed\_maintenance)=Yes));

| **Query8** | | | |
| --- | --- | --- | --- |
| **Make/Model** | **Vendor's name** | **Last Maintained** | **Maintenance Needed?** |
| 4564298 | Coke | 4/4/2016 | Yes |
| 2A-35S | make-it machinery | 4/1/2016 | Yes |

The results are as follows:

The results display the information about the equipment that needs maintenance.

Query 9: This query has the user enter a date to see if the equipment was/is rented out for that day and what time it is being rented out for by accessing the EQUIPMENT table, displaying the equipment make/model, rent out date, and the rent out time. The SQL code is as follows:

SELECT EQUIPMENT.make\_model, EQUIPMENT.rent\_out\_date, EQUIPMENT.rent\_out\_time

FROM EQUIPMENT

WHERE (((EQUIPMENT.rent\_out\_date)=[Enter a date]));

| **Query9** | | |
| --- | --- | --- |
| **make\_model** | **rent\_out\_date** | **rent\_out\_time** |
| 2A-34S | 4/27/2016 | 3:40pm |
| 2A-36S | 4/27/2016 | 6:30pm |

The results for 4/27/2016 are as follows:

The results depend on the date entered. If the date 4/27/2016 is entered, two entries are displayed based off of the previously entered date.

Query 10: The purpose of this query is to prompt the user to input a class number to search the CLASSES and EMPLOYEES tables to find out the backup instructor for a certain class and display said person’s name, address, and phone number. The SQL is as follows:

SELECT DISTINCT CLASSES.number, CLASSES.backup\_employee, EMPLOYEES.address, EMPLOYEES.employee\_phone

FROM EMPLOYEES INNER JOIN CLASSES ON EMPLOYEES.ename = CLASSES.[EMPLOYEES$ename]

WHERE (((CLASSES.number)=[Enter the class' number]));

The results for class number 500 are as follows:

| **Query10** | | | |
| --- | --- | --- | --- |
| **Class Number** | **Backup Employee** | **Backup Employee Address** | **Backup Employee Phone Number** |
| 500 | Heather Boin | 395 Chipp Ave | 3305932596 |

The results are based off of what class number is entered. In this case, 500 is entered for the class date, which displays the backup employee for that class and their contact information.