**Summary:** My enterprise is a wellness website that has nutritional information for food, and information on the calories burnt for some exercises. A customer can get a daily meal plan created from the list of foods. Customers will be assigned a nutritional consultant, a list of which will be in the database. Customers will also have account information, and the customer accounts will be updated by accountants who work for the website. There will also be food coupons available to the customers.

**Entities (with assumptions and constraints):**

* Consultant: consultantID, lastName, experience

I’m assuming each consultant has a unique consultantID, and experience is in years.

* Account: accountNumber, type, monthlyFee.

I’m assuming each account has a unique accountNumber, and there a few different membership types.

* Accountant: accountantID, lastName, salary

I’m assuming each accountant has a unique accountantID.

* Customer: custID, lastName, firstName, phoneNum, streetAddress, city, state, zip

I’m assuming each customer has a unique custID.

* Exercise: exerciseName, duration, caloriesBurnt

I’m assuming each exercise has a unique name, and duration will be in minutes.

* Coupon (Weak Entity): discount, expiryDate

I’m assuming that for each food (strong entity), the combination of the discount amount and expiryDate uniquely identifies a coupon. These attributes are the discriminator of the entity.

* Food: foodName, servingSize, calories, fat, sodium, fiber, sugar, protein

I’m assuming each food has a unique name.

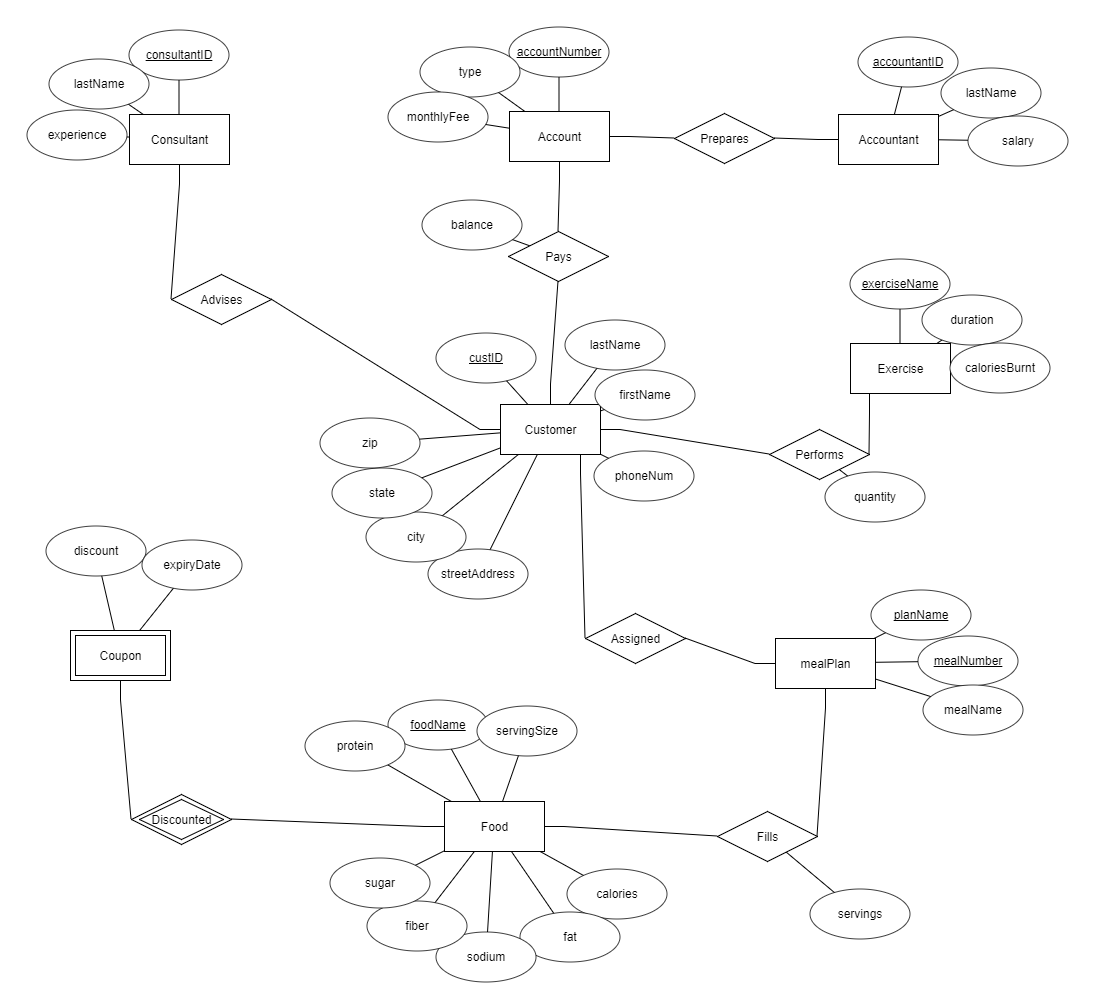
* mealPlan: planName, mealNumber, mealName

The plan name will be something like “vegan” or “Bob’s plan.” The meal number represents which meal of the day it is. The meal name is something like “breakfast” etc. The meal plans are a plan for one meal, so each plan requires both the plan name and meal number to uniquely identify it.

**Relationships (with assumptions, constraints, cardinality, participation):**

* Advises, which is a one-to-many relationship between consultants and customers. A customer has at most one consultant, but may not have been assigned one yet. A consultant may currently have between zero and many customers.
* Pays, which is a one-to-one relationship between customers and accounts. A customer may be a former customer with no current account, so a customer has zero or one accounts. Each account must have exactly one customer connected to it. Balance is an attribute of this relationship, and it represents the current account balance.
* Prepares, which is a one-to-many relationship between accountants and accounts. Each accountant may have between zero and many accounts. Each account must have exactly one accountant who prepares it.
* Performs, which is a many-to-many relationship between customers and exercises. Each customer may perform between zero and many exercises. Each exercise may be performed by between zero and many customers. Quantity is an attribute attached to this relationship which represents the number of sets of this exercise performed by the customer.
* Assigned, which is a many-to-many relationship between customers and meal plans. Each customer is assigned between zero and many meal plans. Each meal plan is assigned to between zero and many customers.
* Fills, which is a many-to-many relationship between food and meal plans. Each food is listed on between zero and many meal plans. Each meal plan contains between zero and many foods. I’m assuming it is possible a meal plan has been named but not yet populated with food. Servings is an attribute of this relationship which describes how many servings of a given food is in a meal plan.
* Discounted, which is a one-to-many relationship between the strong entity food and the weak entity coupon. Each food may have between zero and many coupons associated with it. Each coupon must have exactly one food, on which it is existence-dependent.

The diagram is on the next page. It may be helpful to zoom in.



1

M

M

M

M

M

M

M

1

M

1

1

M

1

**Relational Schema** (primary keys underlined, foreign keys in italics)

Consultant (consultantID, lastName, experience)

Account (accountNumber, type, monthlyFee, balance, *accountantID*)

Accountant (accountantID, lastName, salary)

Customer (custID, lastName, firstName, phoneNum, streetAddress, city, state, zip, *consultantID*,

*accountNumber*)

Exercise (exerciseName, duration, caloriesBurnt)

Food (foodName, servingSize, calories, fat, sodium, fiber, sugar, protein)

mealPlan (planName, mealNumber, mealName)

Coupon (*foodName*, discount, expiryDate)

Performs (*custID*, *exerciseName*, quantity)

Assigned (*custID*, *planName*, *mealNumber*)

Fills (*planName*, *mealNumber*, *foodName*, servings)

**Normalization**

* Consultant (consultantID, lastName, experience)

Functional Dependencies:

consultantID 🡪 lastName, experience

We choose consultantID as primary key.

1NF: Each cell would be single-valued, so Consultant is in 1NF.

2NF: Since we have 1NF and the key consists of a single attribute, we also have 2NF.

3NF: Since no non-key attribute or combination of attributes functionally determines another non-key attribute, we have 3NF.

BCNF: Since we have 3NF with only a single candidate key, we have BCNF.

* Account (accountNumber, type, monthlyFee, balance, *accountantID*)

Functional Dependencies:

accountNumber 🡪 type, monthlyFee, balance, accountantID

type 🡪 monthlyFee

We choose accountNumber as primary key.

1NF: Each cell would be single-valued, so Account is in 1NF.

2NF: Since we have 1NF and the key consists of a single attribute, we also have 2NF.

3NF: We are not in 3NF since we have a transitive dependency, type is not a superkey, and monthlyFee is not part of a candidate key. We need to rewrite account:

Account (accountNumber, *type*, balance, *accountantID*)

Fees (type, monthlyFee)

BCNF: In both relations, the only determinant is the primary key. We are in BCNF.

Preservation: All the original attributes are in some relation, so we have attribute preservation. Since there was no change from 3NF to BCNF, we can see that we have preserved functional dependencies. We have lossless decomposition because we placed the functionally dependent attribute in a relation with its determinant, and kept the determinant itself in the original relation.

* Accountant (accountantID, lastName, salary)

Functional Dependencies:

accountantID 🡪 lastName, salary

We choose accountantID as primary key.

1NF: Each cell would be single-valued, so Accountant is in 1NF.

2NF: Since we have 1NF and the key consists of a single attribute, we also have 2NF.

3NF: Since no non-key attribute or combination of attributes functionally determines another non-key attribute, we have 3NF.

BCNF: Since we have 3NF with only a single candidate key, we have BCNF.

* Customer (custID, lastName, firstName, phoneNum, streetAddress, city, state, zip, *consultantID*, *accountNumber*)

Functional Dependencies:

custID 🡪 lastName, firstName, phoneNum, streetAddress, city, state, zip, consultantID, accountNumber

We choose custID as primary key.

Note: zip does not functionally determine city. From research, there are some cases where the boundaries of a zip code encompass parts of multiple cities. In any case, it is best to leave all the parts of the address in a single relation (for efficiency).

Note: accountNumber is not a candidate key. We are assuming some customers are former customers or potential customers without an account number. Thus, the value can be null, and it is not a determinant.

1NF: Each cell would be single-valued, so Customer is in 1NF.

2NF: Since we have 1NF and the key consists of a single attribute, we also have 2NF.

3NF: Since no non-key attribute or combination of attributes functionally determines another non-key attribute, we have 3NF.

BCNF: Since we have 3NF with only a single candidate key, we have BCNF.

* Exercise (exerciseName, duration, caloriesBurnt)

Functional Dependencies:

exerciseName 🡪 duration, caloriesBurnt

We choose exerciseName as primary key.

Note: exerciseName determines duration in the same sense that foodName determines the serving size. Duration is a “serving” of some exercise. The actual repetitions of this duration of exercise are handled elsewhere.

1NF: Each cell would be single-valued, so Exercise is in 1NF.

2NF: Since we have 1NF and the key consists of a single attribute, we also have 2NF.

3NF: Since no non-key attribute or combination of attributes functionally determines another non-key attribute, we have 3NF.

BCNF: Since we have 3NF with only a single candidate key, we have BCNF.

* Food (foodName, servingSize, calories, fat, sodium, fiber, sugar, protein)

Functional Dependencies:

foodName 🡪 servingSize, calories, fat, sodium, fiber, sugar, protein

We choose foodName as primary key.

1NF: Each cell would be single-valued, so Food is in 1NF.

2NF: Since we have 1NF and the key consists of a single attribute, we also have 2NF.

3NF: Since no non-key attribute or combination of attributes functionally determines another non-key attribute, we have 3NF.

BCNF: Since we have 3NF with only a single candidate key, we have BCNF.

* mealPlan (planName, mealNumber, mealName)

Functional Dependencies:

planName, mealNumber 🡪 mealName

We choose the combination {planName, mealNumber} as primary key.

Note: We don’t have mealNumber 🡪 mealName because some meal plans may have more meals or fewer meals per day. So, “meal 2” does not necessarily equal “lunch” in all cases.

1NF: Each cell would be single-valued, so mealPlan is in 1NF.

2NF: Even though we have a multi-attribute key, there are no partial dependencies. Thus, we have 2NF.

3NF: Since no non-key attribute or combination of attributes functionally determines another non-key attribute, we have 3NF.

BCNF: The only determinant is the primary key. We are in BCNF.

* Coupon (*foodName*, discount, expiryDate)

We only have the key, and there are no non-trivial functional dependencies. The relation is normalized to BCNF.

* Performs (*custID*, *exerciseName*, quantity)

Functional Dependencies:

custID, exerciseName 🡪 quantity

We choose the combination {custID, exerciseName} as primary key.

1NF: Each cell would be single-valued, so Performs is in 1NF.

2NF: Even though we have a multi-attribute key, there are no partial dependencies. Thus, we have 2NF.

3NF: Since no non-key attribute or combination of attributes functionally determines another non-key attribute, we have 3NF.

BCNF: The only determinant is the primary key. We are in BCNF.

* Assigned (*custID*, *planName*, *mealNumber*)

We only have the key, and there are no non-trivial functional dependencies. The relation is normalized to BCNF.

* Fills (*planName*, *mealNumber*, *foodName*, servings)

Functional Dependencies:

planName, mealNumber, foodName 🡪 servings

We choose the combination {planName, mealNumber, foodName} as primary key.

1NF: Each cell would be single-valued, so Fills is in 1NF.

2NF: Even though we have a multi-attribute key, there are no partial dependencies. Thus, we have 2NF.

3NF: Since no non-key attribute or combination of attributes functionally determines another non-key attribute, we have 3NF.

BCNF: The only determinant is the primary key. We are in BCNF.

**Summary**

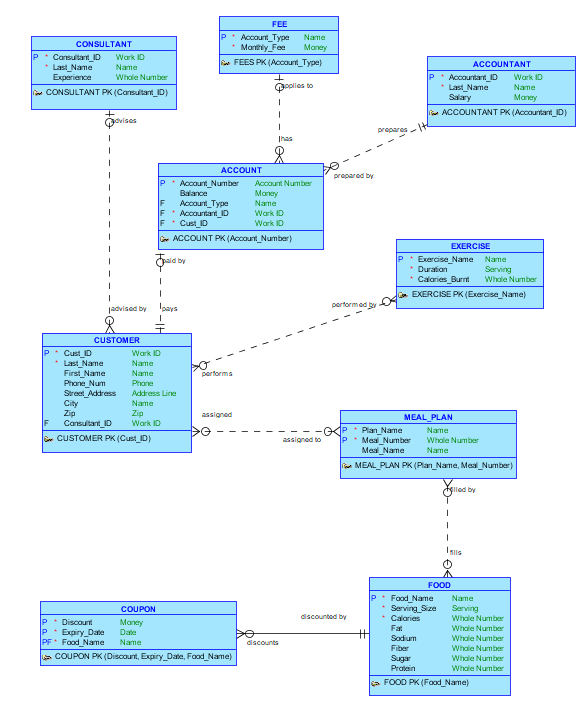
After normalization, we retain the original relational schema, with one exception. The relation

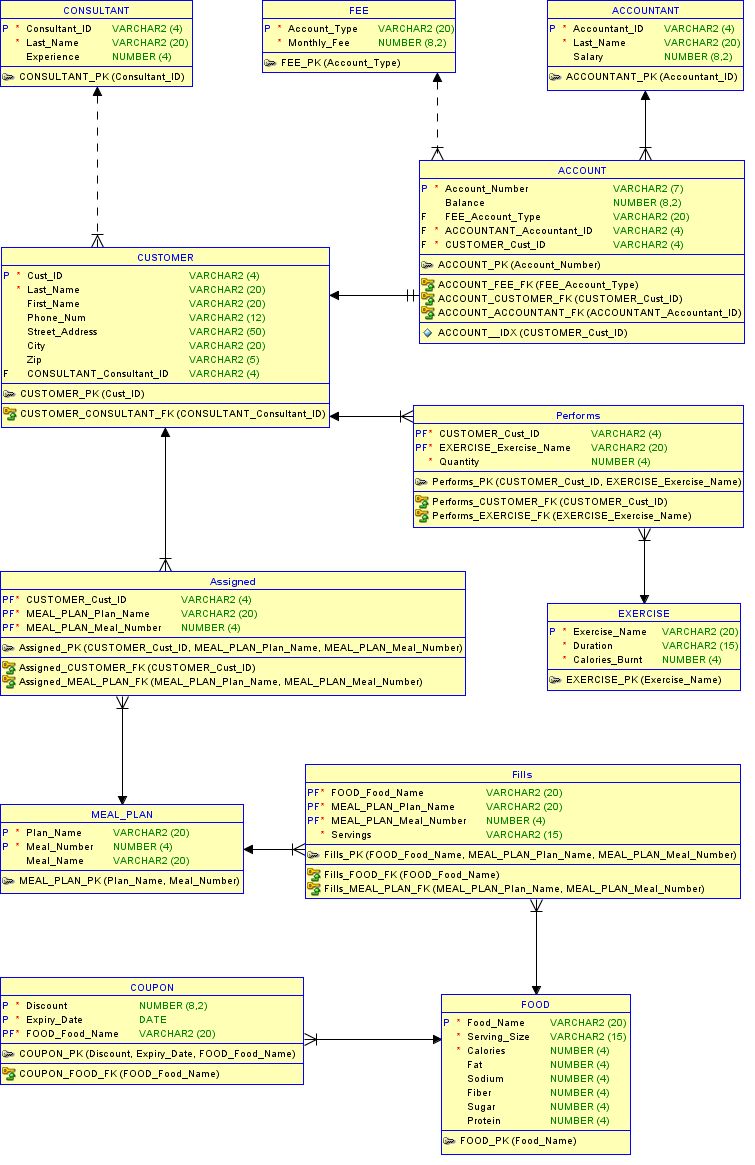
~~Account (accountNumber, type, monthlyFee, balance,~~ *~~accountantID~~*~~)~~ is replaced by

Account (accountNumber, *type*, balance, *accountantID*) and

Fees (type, monthlyFee).

**Contents:** Logical Model (This Page), Relational Model (Next Page), DDL (Page 11-16).





CREATE TABLE account (

account\_number VARCHAR2(7) NOT NULL,

balance NUMBER(8,2),

fee\_account\_type VARCHAR2(20),

accountant\_accountant\_id VARCHAR2(4) NOT NULL,

customer\_cust\_id VARCHAR2(4) NOT NULL

);

CREATE UNIQUE INDEX account\_\_idx ON

account ( customer\_cust\_id ASC );

ALTER TABLE account ADD CONSTRAINT account\_pk PRIMARY KEY ( account\_number );

CREATE TABLE accountant (

accountant\_id VARCHAR2(4) NOT NULL,

last\_name VARCHAR2(20) NOT NULL,

salary NUMBER(8,2)

);

ALTER TABLE accountant ADD CONSTRAINT accountant\_pk PRIMARY KEY ( accountant\_id );

CREATE TABLE assigned (

customer\_cust\_id VARCHAR2(4) NOT NULL,

meal\_plan\_plan\_name VARCHAR2(20) NOT NULL,

meal\_plan\_meal\_number NUMBER(4) NOT NULL

);

ALTER TABLE assigned

ADD CONSTRAINT assigned\_pk PRIMARY KEY ( customer\_cust\_id,

meal\_plan\_plan\_name,

meal\_plan\_meal\_number );

CREATE TABLE consultant (

consultant\_id VARCHAR2(4) NOT NULL,

last\_name VARCHAR2(20) NOT NULL,

experience NUMBER(4)

);

ALTER TABLE consultant ADD CONSTRAINT consultant\_pk PRIMARY KEY ( consultant\_id );

CREATE TABLE coupon (

discount NUMBER(8,2) NOT NULL,

expiry\_date DATE NOT NULL,

food\_food\_name VARCHAR2(20) NOT NULL

);

ALTER TABLE coupon

ADD CONSTRAINT coupon\_pk PRIMARY KEY ( discount,

expiry\_date,

food\_food\_name );

CREATE TABLE customer (

cust\_id VARCHAR2(4) NOT NULL,

last\_name VARCHAR2(20) NOT NULL,

first\_name VARCHAR2(20),

phone\_num VARCHAR2(12),

street\_address VARCHAR2(50),

city VARCHAR2(20),

zip VARCHAR2(5),

consultant\_consultant\_id VARCHAR2(4)

);

ALTER TABLE customer ADD CONSTRAINT customer\_pk PRIMARY KEY ( cust\_id );

CREATE TABLE exercise (

exercise\_name VARCHAR2(20) NOT NULL,

duration VARCHAR2(15) NOT NULL,

calories\_burnt NUMBER(4) NOT NULL

);

ALTER TABLE exercise ADD CONSTRAINT exercise\_pk PRIMARY KEY ( exercise\_name );

CREATE TABLE fee (

account\_type VARCHAR2(20) NOT NULL,

monthly\_fee NUMBER(8,2) NOT NULL

);

ALTER TABLE fee ADD CONSTRAINT fee\_pk PRIMARY KEY ( account\_type );

CREATE TABLE fills (

food\_food\_name VARCHAR2(20) NOT NULL,

meal\_plan\_plan\_name VARCHAR2(20) NOT NULL,

meal\_plan\_meal\_number NUMBER(4) NOT NULL,

servings VARCHAR2(15) NOT NULL

);

ALTER TABLE fills

ADD CONSTRAINT fills\_pk PRIMARY KEY ( food\_food\_name,

meal\_plan\_plan\_name,

meal\_plan\_meal\_number );

CREATE TABLE food (

food\_name VARCHAR2(20) NOT NULL,

serving\_size VARCHAR2(15) NOT NULL,

calories NUMBER(4) NOT NULL,

fat NUMBER(4),

sodium NUMBER(4),

fiber NUMBER(4),

sugar NUMBER(4),

protein NUMBER(4)

);

ALTER TABLE food ADD CONSTRAINT food\_pk PRIMARY KEY ( food\_name );

CREATE TABLE meal\_plan (

plan\_name VARCHAR2(20) NOT NULL,

meal\_number NUMBER(4) NOT NULL,

meal\_name VARCHAR2(20)

);

ALTER TABLE meal\_plan ADD CONSTRAINT meal\_plan\_pk PRIMARY KEY ( plan\_name,

meal\_number );

CREATE TABLE performs (

customer\_cust\_id VARCHAR2(4) NOT NULL,

exercise\_exercise\_name VARCHAR2(20) NOT NULL,

quantity NUMBER(4) NOT NULL

);

ALTER TABLE performs ADD CONSTRAINT performs\_pk PRIMARY KEY ( customer\_cust\_id,

exercise\_exercise\_name );

ALTER TABLE account

ADD CONSTRAINT account\_accountant\_fk FOREIGN KEY ( accountant\_accountant\_id )

REFERENCES accountant ( accountant\_id );

ALTER TABLE account

ADD CONSTRAINT account\_customer\_fk FOREIGN KEY ( customer\_cust\_id )

REFERENCES customer ( cust\_id );

ALTER TABLE account

ADD CONSTRAINT account\_fee\_fk FOREIGN KEY ( fee\_account\_type )

REFERENCES fee ( account\_type );

ALTER TABLE assigned

ADD CONSTRAINT assigned\_customer\_fk FOREIGN KEY ( customer\_cust\_id )

REFERENCES customer ( cust\_id );

ALTER TABLE assigned

ADD CONSTRAINT assigned\_meal\_plan\_fk FOREIGN KEY ( meal\_plan\_plan\_name,

meal\_plan\_meal\_number )

REFERENCES meal\_plan ( plan\_name,

meal\_number );

ALTER TABLE coupon

ADD CONSTRAINT coupon\_food\_fk FOREIGN KEY ( food\_food\_name )

REFERENCES food ( food\_name );

ALTER TABLE customer

ADD CONSTRAINT customer\_consultant\_fk FOREIGN KEY ( consultant\_consultant\_id )

REFERENCES consultant ( consultant\_id );

ALTER TABLE fills

ADD CONSTRAINT fills\_food\_fk FOREIGN KEY ( food\_food\_name )

REFERENCES food ( food\_name );

ALTER TABLE fills

ADD CONSTRAINT fills\_meal\_plan\_fk FOREIGN KEY ( meal\_plan\_plan\_name,

meal\_plan\_meal\_number )

REFERENCES meal\_plan ( plan\_name,

meal\_number );

ALTER TABLE performs

ADD CONSTRAINT performs\_customer\_fk FOREIGN KEY ( customer\_cust\_id )

REFERENCES customer ( cust\_id );

ALTER TABLE performs

ADD CONSTRAINT performs\_exercise\_fk FOREIGN KEY ( exercise\_exercise\_name )

REFERENCES exercise ( exercise\_name );

**Table Data**

Account:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACCOUNT\_NUMBER | BALANCE | ACCOUNT\_TYPE | ACCOUNTANT\_ID | CUST\_ID |
| BTA0011 | 20 | BRONZE | A001 | U001 |
| STA0021 | 0 | SILVER | A001 | U002 |
| GTA0032 | 0 | GOLD | A002 | U003 |
| GTA0042 | 50 | GOLD | A002 | U004 |
| PTA0053 | 70 | PLATINUM | A003 | U005 |

Accountant:

|  |  |  |
| --- | --- | --- |
| ACCOUNTANT\_ID | LAST\_NAME | SALARY |
| A001 | Johnson | 50000 |
| A002 | McHenry | 40000 |
| A003 | Sanders | 30000 |

Assigned:

|  |  |  |
| --- | --- | --- |
| CUST\_ID | PLAN\_NAME | MEAL\_NUMBER |
| U001 | VEGAN | 1 |
| U001 | VEGAN | 2 |
| U001 | VEGAN | 3 |
| U001 | VEGAN | 4 |
| U001 | VEGAN | 5 |
| U002 | FAST FOOD | 3 |
| U003 | FAST FOOD | 1 |
| U003 | FAST FOOD | 2 |
| U003 | FAST FOOD | 3 |
| U004 | VEGAN | 4 |
| U005 | VEGAN | 5 |

Consultant:

|  |  |  |
| --- | --- | --- |
| CONSULTANT\_ID | LAST\_NAME | EXPERIENCE |
| C001 | Mathews | 7 |
| C002 | Wilkins | 10 |
| C003 | Snyder | 3 |

Coupon:

|  |  |  |
| --- | --- | --- |
| DISCOUNT | EXPIRY\_DATE | FOOD\_NAME |
| 75 | 1-Jan-18 | BREAD |
| 50 | 2-Feb-18 | APPLE |
| 1.5 | 31-Dec-17 | RICE |
| 1 | 1-Jan-19 | PEANUT BUTTER |
| 1 | 1-Jun-18 | SOY MILK |
| 75 | 1-Mar-18 | BEANS |
| 50 | 1-Jan-20 | RICE |

Customer:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CUST\_ID | LAST\_NAME | FIRST\_NAME | PHONE\_NUM | STREET\_ADDRESS | CITY | ZIP | CONSULTANT\_ID |
| U001 | SMITH | LARRY | 730-912-5512 | 67 5th Avenue | Dayton | 45424 | C001 |
| U002 | DOE | JANE | 419-884-9979 | 110 1st Street | Toledo | 49983 | C002 |
| U003 | EMERSON | RALPH | 614-989-2399 | 1220 Elm Street | Columbus | 49328 | C001 |
| U004 | WILLIAMS | MISTY | 730-928-2989 | 221 Baker Street | Springfield | 45214 | C003 |
| U005 | BIG | MISTER | 730-893-9298 | 515 Roady Road | Kettering | 45211 | C003 |

Exercise:

|  |  |  |
| --- | --- | --- |
| EXERCISE\_NAME | DURATION | CALORIES\_BURNT |
| RUNNING | 30 minutes | 400 |
| CYCLING | 60 minutes | 600 |
| CRUNCHES | 20 minutes | 150 |
| CURLS | 15 minutes | 100 |
| PUSHUPS | 10 minutes | 125 |
| PULLUPS | 10 minutes | 125 |
| STRETCHING | 15 minutes | 75 |
| SWIMMING | 20 minutes | 300 |

Fee:

|  |  |
| --- | --- |
| ACCOUNT\_TYPE | MONTHLY\_FEE |
| BRONZE | 4.99 |
| SILVER | 9.99 |
| GOLD | 14.99 |
| PLATINUM | 19.99 |

Fills:

|  |  |  |  |
| --- | --- | --- | --- |
| FOOD\_NAME | PLAN\_NAME | MEAL\_NUMBER | SERVINGS |
| PEAS | VEGAN | 1 | 3 |
| BREAD | VEGAN | 1 | 1 |
| APPLE | VEGAN | 1 | 1 |
| SPINACH | VEGAN | 2 | 3 |
| BREAD | VEGAN | 2 | 1 |
| PEANUT BUTTER | VEGAN | 2 | 1 |
| JAM | VEGAN | 2 | 1 |
| CARROTS | VEGAN | 2 | 1 |
| SOY MILK | VEGAN | 3 | 1 |
| BANANA | VEGAN | 3 | 1 |
| BEANS | VEGAN | 4 | 2 |
| RICE | VEGAN | 4 | 1 |
| TOMATOES | VEGAN | 4 | 1 |
| NUTS | VEGAN | 4 | 1 |
| BREAD | VEGAN | 5 | 2 |
| APPLE | VEGAN | 5 | 1 |
| PANCAKES | FAST FOOD | 1 | 2 |
| WHOPPER | FAST FOOD | 2 | 1 |
| FRIES | FAST FOOD | 2 | 1 |
| FRIED CHICKEN | FAST FOOD | 3 | 3 |
| BIG MAC | FAST FOOD | 3 | 2 |
| PANCAKES | FAST FOOD | 3 | 2 |

Food:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| FOOD\_NAME | SERVING\_SIZE | CALORIES | FAT | SODIUM | FIBER | SUGAR | PROTEIN |
| CARROTS | 6 baby | 35 | 0 | 0 | 2 | 0 | 1 |
| TOMATOES | 1 cup | 20 | 0 | 0 | 1 | 0 | 1 |
| SPINACH | 1/2 cup | 30 | 0 | 90 | 3 | 0 | 4 |
| APPLE | 1 large | 110 | 0 | 0 | 3 | 0 | 2 |
| BANANA | 1 medium | 105 | 0 | 0 | 1 | 0 | 2 |
| BREAD | 1 slice | 100 | 2 | 100 | 3 | 2 | 4 |
| PEAS | 1/2 cup | 60 | 0 | 0 | 2 | 0 | 4 |
| SOY MILK | 1 cup | 110 | 4 | 85 | 0 | 5 | 7 |
| PEANUT BUTTER | 2 tbsp | 190 | 18 | 100 | 3 | 3 | 7 |
| JAM | 1 tbsp | 40 | 0 | 10 | 1 | 8 | 0 |
| BEANS | 1/2 cup | 110 | 0 | 0 | 3 | 0 | 4 |
| RICE | 1/2 cup | 180 | 2 | 0 | 3 | 0 | 4 |
| NUTS | 1 oz | 170 | 17 | 90 | 3 | 0 | 7 |
| BIG MAC | 1 sandwich | 563 | 40 | 1000 | 3 | 5 | 30 |
| FRIES | 1 large | 365 | 20 | 800 | 3 | 4 | 8 |
| WHOPPER | 1 sandwich | 677 | 50 | 1300 | 4 | 5 | 40 |
| FRIED CHICKEN | 1 breast | 161 | 20 | 400 | 1 | 4 | 35 |
| PANCAKES | 3 cakes | 192 | 8 | 200 | 2 | 20 | 8 |

Meal Plan:

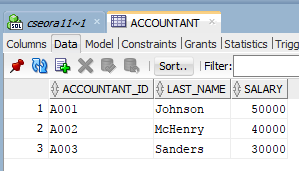
|  |  |  |
| --- | --- | --- |
| PLAN\_NAME | MEAL\_NUMBER | MEAL\_NAME |
| VEGAN | 1 | BREAKFAST |
| VEGAN | 2 | BRUNCH |
| VEGAN | 3 | SNACK |
| VEGAN | 4 | LUNCH |
| VEGAN | 5 | DINNER |
| FAST FOOD | 1 | BREAKFAST |
| FAST FOOD | 2 | LUNCH |
| FAST FOOD | 3 | DINNER |

Performs:

|  |  |  |
| --- | --- | --- |
| CUST\_ID | EXERCISE\_NAME | QUANTITY |
| U001 | RUNNING | 1 |
| U001 | CRUNCHES | 1 |
| U002 | CYCLING | 2 |
| U003 | SWIMMING | 2 |
| U003 | PUSHUPS | 1 |
| U004 | STRETCHING | 1 |

**SQL QUERIES**

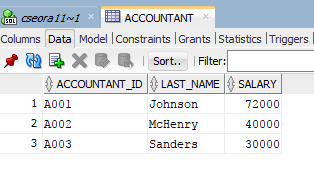
1. Update Example



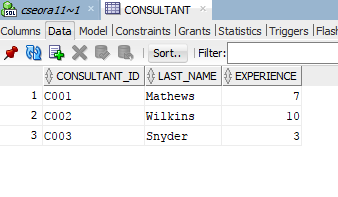
UPDATE Accountant

SET salary = salary + 22000

WHERE Accountant\_ID = 'A001';



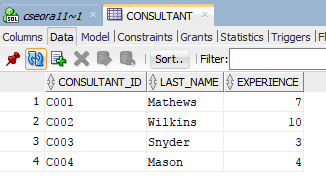
1. Insert Example



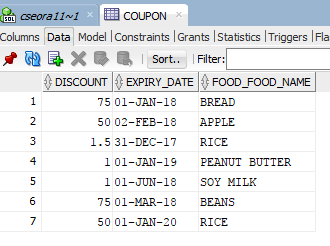
INSERT

INTO Consultant

VALUES ('C004', 'Mason', 4);



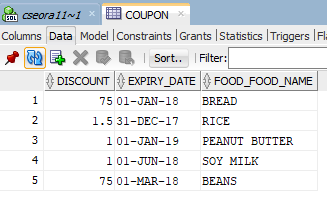
1. Delete Example



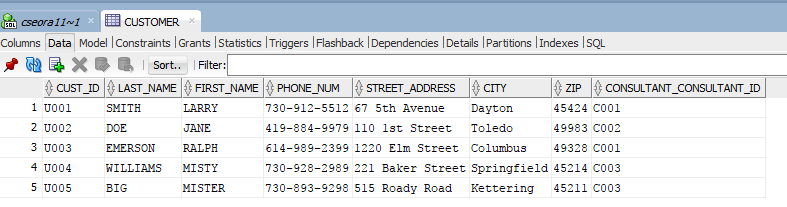
DELETE

FROM Coupon

WHERE Discount = 50;



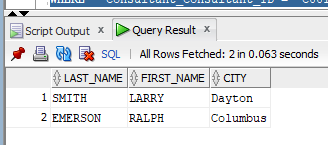
1. Select Example (single table)



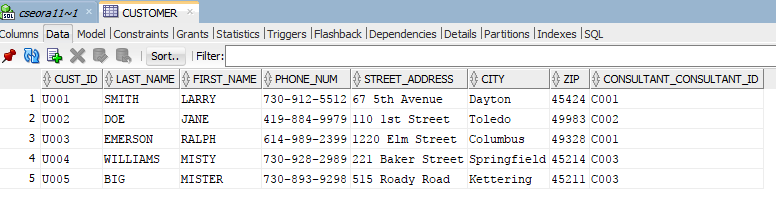
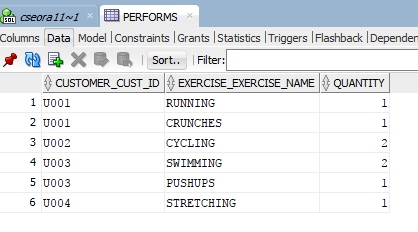
SELECT last\_name, first\_name, city

FROM Customer

WHERE Consultant\_Consultant\_ID = 'C001';



1. Select Example (multiple tables):

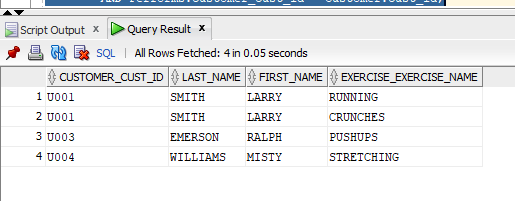


SELECT Performs.customer\_cust\_id, last\_name, first\_name, Performs.exercise\_exercise\_name

FROM Performs, Customer

WHERE quantity = 1

AND Performs.customer\_cust\_id = Customer.cust\_id;



**Conclusion**

I learned a lot about database concepts by doing this project. I feel much more prepared to tackle issues of ER diagrams and normalization of relational schemas. I was happy to get some practical experience with the Oracle products we used this semester: Data Modeler and SQL Developer. I also learned more about putting reports together for a project like this. I was glad that we developed the project in successive steps over the course of the semester. I initially had very ambitious plans to build a massive food database with lots of accurate information, but I realized that simply plugging in data is not exactly a learning experience, nor is it a good use of my time. I enjoyed completing this project!