```
/* ****************************************************************************
          INTRODUCTION TO STRUCTURED QUERY LANGUAGE FOR DATA ANALYTICS
                    SF25SQL6172025, 2025/11/17 - 2025/12/17
                  https://folvera.commons.gc.cuny.edu/?cat=35
 ****************************************************************************

SESSION #1 (2025/11/17): COOKIES & MILK NOW, CODE LATER

 1. For this example with cookies, we first need to create the data.  At this
    point of the course, you should not be concerned about the code, but rather
    what the language can do.  We will have fun writing code starting on day 2.

    1.1. On the right side, there is a quick explanation what each line (part
         of a single statement) does.

    1.2. Note that a statement starts with an English verb and ends with a
         semicolon.
 **************************************************************************** */

CREATE DATABASE food_stores;                        -- 1. creating database of food
                                                    --    stores (`food_stores`)
CREATE SCHEMA nyc;                                  -- 2. creating schema for food
                                                    --    stores in NYC (`nyc`)
CREATE TABLE nyc.snack_store (                      -- 3. creating table for stores
  record_id INT NOT NULL,                           --    where we are shopping
  store_id INT,                                     --    (`snack_store`) in schema
  snack_id INT,                                     --    `nyc`; in other words,
  food_type VARCHAR(50),                            --    another food store in NYC
  flavor VARCHAR(50),
  organic VARCHAR(1),
  dietary_restriction VARCHAR(50),
  package_size VARCHAR(50),
  brand_name VARCHAR(50)
  );

INSERT INTO nyc.snack_store                         -- 4. inserting values in table
VALUES (                                            --    `nyc.snack_store`
  1,
  1,
  1,
  'cookie',
  'chocolate',
  'y',
  'gluten free',
  'small',
  'Good Cookies'
  ),
  (
  2,
  1,
  1,
  'cookie',
  'chocolate',
  'y',
  '',
  'small',
  'Good Cookies'
  ),
  (
  3,
  1,
  1,
  'cookie',
  'chocolate',
  'y',
  'nut free',
  'small',
  'Good Cookies'
  ),
```

```
    (
    4,
    1,
    2,
    'donut',
    'chocolate',
    'y',
    'gluten free',
    'small',
    'Good Cookies'
    ),
    (
    5,
    2,
    1,
    'cookie',
    'chocolate',
    'y',
    '',
    'small',
    'Good Cookies'
    ),
    (
    6,
    2,
    1,
    'cookie',
    'vanilla',
    'y',
    'gluten free',
    'large',
    'Cool Snacks'
    ),
    (
    7,
    2,
    1,
    'cookie',
    'vanilla',
    'y',
    'nut free',
    'single serving',
    'Cool Snacks'
    ),
    (
    8,
    1,
    1,
    'cookie',
    'chocolate',
    'n',
    'gluten free',
    'small',
    'Cookies by Alice'
    ),
    (
    9,
    3,
    1,
    'cookie',
    'chocolate',
    'n',
    'gluten free',
    'small',
    'Cookies by Alice'
    ),
    (
    10,
    3,
```

```
    1,
    'cookie',
    'chocolate',
    'y',
    'gluten free',
    'small',
    'Good Cookies'
    ),
    (
    11,
    3,
    1,
    'cookie',
    'vanilla',
    'y',
    'gluten free',
    'small',
    'Cool Snacks'
    ),
    (
    12,
    1,
    2,
    'donut',
    'jelly',
    'y',
    'gluten free',
    'small',
    'Good Cookies'
    ),
    (
    13,
    1,
    1,
    'cookie',
    'chocolate',
    'n',
    'gluten free',
    'small',
    'Cookies by Alice'
    ),
    (
    14,
    3,
    1,
    'cookie',
    'chocolate',
    'n',
    'gluten free',
    'small',
    'Cookies by Alice'
    ),
    (
    15,
    3,
    1,
    'cookie',
    'chocolate',
    'n',
    'gluten free',
    'small',
    'Cookies by Alice'
    ),
    (
    16,
    3,
    1,
    'cookie',
    'chocolate',
```

```
    'y',
    '',
    'large',
    'Grand Cookies'
    ),
    (
    17,
    3,
    1,
    'cookie',
    'vanilla',
    'y',
    'vegan',
    'large',
    'Good Cookies'
    ),
    (
    18,
    3,
    1,
    'cookie',
    'vanilla',
    'y',
    'vegan',
    'large',
    'Good Cookies'
    ),
    (
    19,
    3,
    1,
    'cookie',
    'chocolate',
    'y',
    'gluten free',
    'small',
    'Grand Cookies'
    );

CREATE TABLE nyc.store_names (
  store_id INT NOT NULL,
  store_name VARCHAR(50)
  );

INSERT INTO nyc.store_names
VALUES (
  1,
  'The Nature Shop'
  ),
  (
  2,
  'Supermarket Cool Banana'
  ),
  (
  3,
  'The Corner Shop'
  );

SELECT *
FROM nyc.store_names;


/* ********************************************************************
    1.3. In this example, you are an assistant to a boss who wants cookies from
         `The Corner Shop`.

         We will keep adding line by line to redefine what we need.  As you can
         see, the English request is translated to SQL in an English-like
         series of expressions.
```

```
277
278           Note that, if you read the statement out loud and it does not sound
279           right as an English imperative sentence (a command, in this case given
280           to a computer), most likely your syntax is wrong.
281    ********************************************************************** */
282
283    SELECT *                                          -- 1. getting all information
284    FROM nyc.snack_store                              -- 2. from specific population
285                                                      --    `nyc.snack_store`
286    WHERE food_type = 'cookie'                        -- 3. where specific
287                                                      --    `food_type` is `cookie`
288      AND store_id = 3;                               --    `store_id` is `3`, the
289                                                      --    identifier for `The
290                                                      --    Corner Shop`
291
292
293    /* ********************************************************************
294        1.4. You go to `The Corner Shop` and bring multiple cookies including
295             doubles.  Your boss then says that you should not bring doubles.
296     ********************************************************************** */
297
298    SELECT DISTINCT store_id,                         -- `DISTINCT` = no doubles,
299      snack_id,                                       -- unique cookies, excluding
300      food_type,                                      -- the row identifier
301      flavor,
302      organic,
303      dietary_restriction,
304      package_size,
305      brand_name
306    FROM nyc.snack_store
307    WHERE food_type = 'cookie'
308      AND store_id = 3;
309
310
311    /* ********************************************************************
312        1.5. You go back to the store and bring back one of each cookie.  Then your
313             boss says that he/she wants chocolate cookies.
314     ********************************************************************** */
315
316    SELECT DISTINCT store_id,                         -- 1. still looking for all
317      snack_id,                                       --    information
318      food_type,
319      flavor,
320      organic,
321      dietary_restriction,
322      package_size,
323      brand_name
324    FROM nyc.snack_store                              -- 2. from specific population
325                                                      --    `nyc.snack_store`
326    WHERE food_type = 'cookie'                        -- 3. still looking for a
327                                                      --    `food_type` of `cookie`
328      AND store_id = 3                                --    and `store_id` 3
329      AND flavor = 'chocolate';                       -- 4. new column/value
330                                                      --    `flavor` is `chocolate`
331                                                      -- 5. hence looking for a
332                                                      --    chocolate cookies
333
334
335    /* ********************************************************************
336        1.6. You go back to the store and bring back only chocolate cookies.  Then
337             your boss says that the cookies must be organic.
338     ********************************************************************** */
339
340    SELECT DISTINCT store_id,
341      snack_id,
342      food_type,
343      organic,
344      dietary_restriction,
345      package_size,
```

```
  brand_name
FROM nyc.snack_store
WHERE food_type = 'cookie'
  AND store_id = 3
  AND flavor = 'chocolate'
  AND organic = 'y';


/* ***************************************************************************
    1.7. You go back to the store and bring back only organic chocolate
         cookies.  Then your boss says that the cookies should also be
         gluten-free.
 *************************************************************************** */

SELECT DISTINCT store_id,
  snack_id,
  food_type,
  organic,
  dietary_restriction,
  package_size,
  brand_name
FROM nyc.snack_store
WHERE food_type = 'cookie'
  AND store_id = 3
  AND flavor = 'chocolate'
  AND organic = 'y'
  AND dietary_restriction = 'gluten free';


/* ***************************************************************************
    1.8. You go back to the store and bring back only gluten-free and organic
         chocolate cookies.  Then your boss says the package is small.
 *************************************************************************** */

SELECT DISTINCT store_id,
  snack_id,
  food_type,
  organic,
  dietary_restriction,
  package_size,
  brand_name
FROM nyc.snack_store
WHERE food_type = 'cookie'
  AND store_id = 3
  AND flavor = 'chocolate'
  AND organic = 'y'
  AND dietary_restriction = 'gluten free'
  AND package_size = 'small';


/* ***************************************************************************
    1.9. You go back to the store and bring back only small packages of
         gluten-free and organic chocolate cookies.  Then your boss says that
         the brand name should be `Good Cookies`.
 *************************************************************************** */

SELECT DISTINCT store_id,
  snack_id,
  food_type,
  organic,
  dietary_restriction,
  package_size,
  brand_name
FROM nyc.snack_store
WHERE food_type = 'cookie'
  AND store_id = 3
  AND flavor = 'chocolate'
  AND organic = 'y'
  AND dietary_restriction = 'gluten free'
```

```
415     AND package_size = 'small'
416     AND brand_name = 'Good Cookies';
417
418
419   /* ***************************************************************************
420       1.10. You go back to the store and bring back only small packages of `Good
421             Cookies` gluten-free and organic chocolate cookies.  At this point,
422             your boss is finally satisfied.
423
424             Since the values of `cookies` and `donuts` are repeated in several
425             records (rows), these values could be moved to another table and
426             referred to with a unique identifier (a row ID).  The values of
427             `chocolate` and `vanilla` can also be moved to another table.
428
429             This is the concept of relational databases where tables have unique
430             values that can be referred by unique identifiers -- for example,
431             `food_type_id` for `cookies` and `donuts` and `flavor_id` for
432             `chocolate and `vanilla`.
433
434       1.11. Now that you have a general idea of what SQL does, we can start to
435             look at the code.
436
437    ***************************************************************************
438    https://folvera.commons.gc.cuny.edu/?p=1321
439    *************************************************************************** */
```