```
/* ****************************************************************************
        CUNY ACE UPSKILLING:  INTRODUCTION TO STRUCTURED QUERY LANGUAGE
                      SF21JOB#2, 2021/11/08 to 2021/12/13
                  https://folvera.commons.gc.cuny.edu/?cat=30
 ******************************************************************************

  SESSION #2 (2021/11/10): UNDERSTANDING CORE DATABASE CONCEPTS

  1. Learning history of SQL and basic concepts of the structure of a
     relational database
  2. Understanding structured programming
  3. Understanding naming convention
  4. Understanding basic syntax to query one table
 ******************************************************************************

 1. Database professionals in the labor economy is on the rise.  By 2024, the
    US Bureau of Labor Statistics has projected an 18.8% job increase for
    `Software developers, applications` with a median annual income of $100,080
    and 20.9% for `Computer systems analysts` with a median annual income of
    $87,220 as of 04/14/2017 (https://bls.gov/emp/ep_table_104.htm) creating a
    surge for individuals who possess the right skills to store and query large
    data sets.

    +-----------------------------------------+---------------+---------------+
    | Computer Systems Analysts               | $88,270 / yr  | $42.44 / hr   |
    +-----------------------------------------+---------------+---------------+
    | https://bls.gov/ooh/computer-and-information-technology/computer-systems-analysts.htm
    +-----------------------------------------+---------------+---------------+
    | Database Administrators                 | $87,020 / yr  | $41.84 / hr   |
    +-----------------------------------------+---------------+---------------+
    | https://bls.gov/ooh/computer-and-information-technology/database-administrators.htm
    +-----------------------------------------+---------------+---------------+
    | Web Developers                          | $67,990 / yr  | $32.69 / hr   |
    +-----------------------------------------+---------------+---------------+
    | https://bls.gov/ooh/computer-and-information-technology/web-developers.htm
    +-----------------------------------------+---------------+---------------+
    | Operations Research Analysts            | $81,390 / yr  | $39.13 / hr   |
    +-----------------------------------------+---------------+---------------+
    | https://bls.gov/ooh/math/operations-research-analysts.htm               |
    +-------------------------------------------------------------------------+

 2. The following are concepts that you need to know.

    2.1. SQL (Structured Query Language) is a standardized programming language
         used for managing relational databases and performing various
         operations on the data in them.  Initially created in the 1970s, SQL
         is regularly used by database administrators, as well as by developers
         writing data integration scripts and data analysts looking to set up
         and run analytical queries.
         https://searchsqlserver.techtarget.com/definition/SQL

         The SQL programming language was first developed in the 1970s by IBM
         researchers Raymond Boyce and Donald Chamberlin.  The programming
         language, known then as SEQUEL, was created following the publishing
         of Edgar Frank Todd's paper, ``A Relational Model of Data for Large
         Shared Data Banks,`` in 1970.
         https://businessnewsdaily.com/5804-what-is-sql.html

         Refer to https://ibm.com/ibm/history/ibm100/us/en/icons/reldb/ for
         more information on Edgar Frank Todd's paper.

    2.2. T-SQL (Transact-SQL) is a set of programming extensions from Sybase
         and Microsoft that add several features to the Structured Query
         Language (SQL), including transaction control, exception and error
         handling, row processing and declared variables.
         https://searchsqlserver.techtarget.com/definition/T-SQL

    2.3. Microsoft SQL Server is a relational database management system, or
         RDBMS, that supports a wide variety of transaction processing,
         business intelligence and analytics applications in corporate IT
         environments.  It's one of the three market-leading database
         technologies, along with Oracle Database and IBM's DB2.
         https://searchsqlserver.techtarget.com/definition/SQL-Server

    2.4. A server is a computer program that provides a service to another
         computer programs (and its user).  In a data center, the physical
         computer that a server program runs in is also frequently referred to
         as a server.  That machine may be a dedicated server or it may be used
         for other purposes as well.
         In the client/server programming model, a server program awaits and
         fulfills requests from client programs, which may be running in the
         same or other computers.  A given application in a computer may
         function as a  client with requests for services from other programs
         and also as a server of requests from other programs.
         https://whatis.techtarget.com/definition/server

    2.5. A relational database management system (RDBMS) is a collection of
         programs and capabilities that enable IT teams and others to create,
         update, administer and otherwise interact with a relational database.
         Most commercial RDBMSes use Structured Query Language (SQL) to access
         the database, although SQL was invented after the initial development
         of the relational model and is not necessary for its use.
         https://searchdatamanagement.techtarget.com/definition/RDBMS-relational-database-management-system

    2.6. In computer programming, a schema (pronounced SKEE-mah) is the
         organization or structure for a database.  The activity of data
         modeling leads to a schema. (The plural form is schemata.  The term is
         from a Greek word for ``form`` or ``figure.``  Another word from the
         same source is ``schematic.``)  The term is used in discussing both
         relational databases and object-oriented databases.  The term
         sometimes seems to refer to a visualization of a structure and
```

```
          sometimes to a formal text-oriented description.
          https://searchsqlserver.techtarget.com/definition/schema

     2.7. In computer programming, a table is a data structure used to organize
          information, just as it is on paper.  There are many different types
          of computer-related tables, which work in a number of different ways.
          The following are examples of the more common types.
          1) In data processing, a table (also called an array) is a organized
             grouping of fields.  Tables may store relatively permanent data, or
             may be frequently updated.  For example, a table contained in a
             disk volume is updated when sectors are being written.
          2) In a relational database, a table (sometimes called a file)
             organizes the information about a single topic into rows and
             columns.  For example, a database for a business would typically
             contain a table for customer information, which would store
             customers' account numbers, addresses, phone numbers, and so on as
             a series of columns.  Each single piece of data (such as the
             account number) is a field in the table. A column consists of all
             the entries in a single field, such as the telephone numbers of all
             the customers.  Fields, in turn, are organized as records, which
             are complete sets of information (such as the set of information
             about a particular customer), each of which comprises a row.  The
             process of normalization determines how data will be most
             effectively organized into tables.
          https://whatis.techtarget.com/definition/table

  3. Before we start, you should be familiar with the naming convention used in
     T-SQL (https://searchsqlserver.techtarget.com/definition/T-SQL) using the
     database for this course.

                     PC12345\MSSQLSERVER        (server; name depending on the
                      |                         machine you are using where
                      |                         `PC12345` is the `HOSTNAME` and
                      |                         `MSSQLSERVER` is the database
                      |                         instance)
                      |
                      +- SF21JOB2               (database in server
                           |                    `PC12345\SQLSERVEREXPRESS`)
                           |
                           +- AP1               (schema in database
                               |                `SF21JOB2`)
                               |
                               +- ContactUpdates (table in schema `AP1`)
                                   |
                                   +- VendorID   (column in table
                                                `ContactUpdates`)

     3.1. Using the structure above, `SF21JOB2` is the database
          (https://searchsqlserver.techtarget.com/definition/database).  A
          database (DB) is a collection of related data like schemata, tables,
          views, functions, procedures and other related objects.

```

```sql
    3.2. `AP1` (`SF21JOB2.AP1`) is a schema
         (https://searchsqlserver.techtarget.com/definition/schema) in database
         `SF21JOB2`.  A schema is a collection of tables, views, functions
         and other related objects often used for organizational purposes only.

    3.3. `ContactUpdates` (`SF21JOB2.AP1.ContactUpdates`) is a table
         (https://whatis.techtarget.com/definition/table) in schema `AP1`
         calling the schema because the schema is not `dbo` (`database owner`
         default schema in T-SQL, which does not need to be called when used).
         A table is a collection of columns/fields and rows/records.

    3.4. `VendorID` (`SF21JOB2.AP1.ContactUpdates.VendorID`) is a column/field
         (https://searchoracle.techtarget.com/definition/field) in table
         `AP1.ContactUpdates`.  A column/field is an allocation of data in a
         record/row.

         This column stores the row identifier for the table.

         It is best practice for a row identifier (usually an integer, a whole
         number) to be a unique identifier, preferably not related to the rest
         of the data in the row.

    3.5. A record/row (https://searchoracle.techtarget.com/definition/record)
         is a collection of related data
         (https://searchdatamanagement.techtarget.com/definition/data), not
         referred to with a name but rather its row identifier or position in
         the table.

 4. In order to retrieve data, we use a `SELECT` statement where the simplest
    syntax is the following.

                    SELECT field1, field2 ...
                    FROM table1;

    4.1. In the example below, we retrieve all columns (fields) and all rows
         (records) from `AP1.ContactUpdates` calling each one of the columns.
 *************************************************************************** */

SELECT VendorID,
  VendorName,
  VendorAddress1,
  VendorAddress2,
  VendorCity,
  VendorState,
  VendorZipCode,
  VendorPhone,
  VendorContactLName,
  VendorContactFName,
  DefaultTermsID,
  DefaultAccountNo
FROM AP1.ContactUpdates;
```

```

/* *****************************************************************************
    4.2. In the example below, we retrieve all columns (fields) and all rows
         (records) from `AP1.Vendors` using wild card `*` (read as `all`).
 **************************************************************************** */

SELECT *                                        -- read as `all`
FROM AP1.Vendors;


/* *****************************************************************************
    4.3. In the example below, we retrieve all columns and rows from tables
         `AP1.ContactUpdates` and `AP1.Vendors` using wild card `*` (read as
         `all`).

         Since we are calling a second table, we have to `JOIN` them on the
         common field (data, value) as this indicates the relation between the
         two tables.

         We are going to cover three `JOIN` alternatives.  Each `JOIN` returns
         a different population.

         `INNER JOIN` returns shared data (rows) between the two tables.  Any
         data found only in one table is not returned.

         `LEFT [OUTTER] JOIN` returns all the data (rows) from the left table
         (first table called, `AP1.ContactUpdates`) and any shared data (rows)
         in the right table (second table called, `AP1.Vendors`).  No data is
         ignored.

         `RIGHT [OUTTER] JOIN` returns all the data (rows) from the right table
         (second table called, `AP1.Vendors`) and any shared data (rows) in the
         left table (first table called, `AP1.ContactUpdates`).  Note that this
         may be confusing for anyone reading your code.  You might want to
         avoid using `RIGHT JOIN`.

         4.3.1. As mentioned, in the example below, we retrieve all shared data
                (rows) from tables `AP1.ContactUpdates` and `AP1.Vendors`.
 **************************************************************************** */

SELECT *
FROM AP1.ContactUpdates                         -- 1. all shared data (rows)
                                                --    from `AP1.ContactUpdates`
INNER JOIN AP1.Vendors                          -- 2. all shared data (rows)
                                                --    from `AP1.Vendors`
  ON AP1.ContactUpdates.VendorID = AP1.Vendors.VendorID;
                                                -- 3. on common data (rows)
                                                --    `VendorID`
```

```sql
/* ****************************************************************************
         4.3.2. In the example below, we retrieve all data (rows) from table
                `AP1.ContactUpdates` and any shared data (rows) from
                `AP1.Vendors`.
 *************************************************************************** */

SELECT *
FROM AP1.ContactUpdates                         -- 1. all data (rows) from main
                                                --    table
                                                --    `AP1.ContactUpdates`
LEFT JOIN AP1.Vendors                           -- 2. any shared data (rows)
                                                --    from `AP1.Vendors`
  ON AP1.ContactUpdates.VendorID = AP1.Vendors.VendorID;
                                                -- 3. on common data (rows)
                                                --    `VendorID`


/* ****************************************************************************
         4.3.3. In the example below, we retrieve all data (rows) from table
                `AP1.Vendors` and any shared data (rows) from
                `AP1.ContactUpdates`.
 *************************************************************************** */

SELECT *
FROM AP1.ContactUpdates                         -- 1. any shared data (rows)
                                                --    from `AP1.ContactUpdates`
RIGHT JOIN AP1.Vendors                          -- 2. all data (rows) from main
                                                --    table `AP1.Vendors`
  ON AP1.ContactUpdates.VendorID = AP1.Vendors.VendorID;
                                                -- 3. on common data (rows)
                                                --    `VendorID`


/* ****************************************************************************
    4.4. In the example below, we retrieve all columns (fields) and rows
         (columns) from `AP1.Vendors` calling each one of the columns.

         4.4.1. We use `AS` to assign aliases to ``create a temporary name for
                columns or tables.``

                We can use aliases on columns ``to make column headings in your
                result set easier to read.``

                We can use aliases on tables ``to shorten your SQL to make it
                easier to read or when you are performing a self join (ie:
                listing the same table more than once in the FROM clause).``
                https://techonthenet.com/sql_server/alias.php

         4.4.2. We also use some functions
                (https://techonthenet.com/sql_server/functions/index_alpha.php).

                CONCAT()    allows you to concatenate strings together
```

```sql
                            https://techonthenet.com/sql_server/functions/concat.php
                            `+` also allows you to concatenate strings together
                            although adding NULL returns a NULL
                            https://techonthenet.com/sql_server/functions/concat2.php

                LEN()       returns the length of the specified string... does
                            not include trailing space characters at the end
                            the string when calculating the length
                            https://techonthenet.com/sql_server/functions/len.php

                LTRIM()     removes all space characters from the left-hand
                            side of a string
                            https://techonthenet.com/sql_server/functions/ltrim.php

                LOWER()     converts all letters in the specified string to
                            lowercase
                            https://techonthenet.com/sql_server/functions/lower.php

                REPLACE()   replaces a sequence of characters in a string with
                            another set of characters, not case-sensitive
                            https://techonthenet.com/sql_server/functions/replace.php

                RIGHT()     allows you to extract a substring from a string,
                            starting from the right-most character
                            https://techonthenet.com/sql_server/functions/right.php

                RTRIM()     removes all space characters from the right-hand
                            side of a string
                            https://techonthenet.com/sql_server/functions/rtrim.php

                SUBSTRING() allows you to extract a substring from a string
                            https://techonthenet.com/sql_server/functions/substring.php

                UPPER()     converts all letters in the specified string to
                            uppercase
                            https://techonthenet.com/sql_server/functions/upper.php
 *************************************************************************** */

SELECT VendorID,
  UPPER(VendorName) AS VendorName,              -- 1. using an alias (`AS`)
                                                --    since losing column name
                                                --    with when using function
                                                --    `UPPER()` to make all
```

```
                                                  --    characters lower upper
                                                  --    case
    CONCAT (
      VendorAddress1,
      ' ',
      VendorAddress2
      ) AS VendorAddress,                         -- 2. using an alias (`AS`)
                                                  --    since losing column name
                                                  --    with when using function
                                                  --    `CONCAT()` to concatenate
                                                  --    (to put strings together)
    LOWER(VendorCity) AS VendorCity,              -- 3. using an alias (`AS`)
                                                  --    since losing column name
                                                  --    with when using function
                                                  --    `LOWER()` to make all
                                                  --    characters lower upper
                                                  --    case
    RIGHT(VendorCity, 4) AS VendorCityRight,      -- 4. using an alias (`AS`)
                                                  --    since losing column name
                                                  --    with when using function
                                                  --    `RIGHT()` to retrieve
                                                  --    four (4) characters from
                                                  --    the right
    LEFT(VendorCity, 3) AS VendorCityLeft,        -- 5. using an alias (`AS`)
                                                  --    since losing column name
                                                  --    with when using function
                                                  --    `LEFT()` to retrieve
                                                  --    three (3) characters from
                                                  --    the left
    SUBSTRING(VendorCity, 3, 4) AS VendorCitySubstring,
                                                  -- 6. using an alias (`AS`)
                                                  --    since losing column name
                                                  --    with when using function
                                                  --    `SUBSTRING()` to retrieve
                                                  --    four (4) characters
                                                  --    starting from the the
                                                  --    third (3rd) character
    LEN(VendorCity) AS VendorCityLen,             -- 7. using an alias (`AS`)
                                                  --    since losing column name
                                                  --    with when using function
                                                  --    `LEN()` to retrieve the
                                                  --    length of string in field
    REPLACE(VendorState, 'CA', 'California') AS VendorState,
                                                  -- 8. using an alias (`AS`)
                                                  --    since losing column name
                                                  --    with when using function
                                                  --    `REPLACE()` to replace
                                                  --    string `CA` with string
                                                  --    `California`
    VendorZipCode,
    VendorPhone,
    VendorContactLName AS 'Vendor Contact Last Name',
                                                  -- 9. using an alias (`AS`)
                                                  --    to change the name of
                                                  --    column
    VendorContactFName AS 'Vendor Contact First Name',
                                                  -- 10. using an alias (`AS`)
                                                  --     to change the name of
                                                  --     column
    DefaultTermsID,
    DefaultAccountNo
  FROM AP1.Vendors;


  /* *****************************************************************************
   3. LAB 1
      Write a query calling all shared fields (`INNER JOIN`) from `AP1.Invoices`,
      `AP1.Terms` and `AP1.Vendors`.
      * Delete or rename the duplicate name of the columns.
   **************************************************************************** */

  SELECT AP1.Invoices.InvoiceID,
    AP1.Invoices.VendorID,
    AP1.Invoices.InvoiceNumber,
    AP1.Invoices.InvoiceDate,
    AP1.Invoices.InvoiceTotal,
    AP1.Invoices.PaymentTotal,
    AP1.Invoices.CreditTotal,
    AP1.Invoices.TermsID,
    AP1.Invoices.InvoiceDueDate,
    AP1.Invoices.PaymentDate,
    -- AP1.Terms.TermsID,                         -- 1. duplicate column name
                                                  --    (`TermsID`), which can
                                                  --    be removed (commented
                                                  --    out, in this case)
                                                  --    without affecting the
                                                  --    query output; could also
                                                  --    be renamed
    AP1.Terms.TermsDescription,
    AP1.Terms.TermsDueDays,
    -- AP1.Vendors.VendorID,                      -- 2. duplicate column name
                                                  --    (`VendorID`), which can
                                                  --    be removed (commented
                                                  --    out, in this case)
                                                  --    without affecting the
                                                  --    query output; could also
                                                  --    be renamed
    AP1.Vendors.VendorName,
    AP1.Vendors.VendorAddress1,
    AP1.Vendors.VendorAddress2,
    AP1.Vendors.VendorCity,
    AP1.Vendors.VendorState,
    AP1.Vendors.VendorZipCode,
    AP1.Vendors.VendorPhone,
    AP1.Vendors.VendorContactLName,
    AP1.Vendors.VendorContactFName,
    AP1.Vendors.DefaultTermsID,
    AP1.Vendors.DefaultAccountNo
  FROM AP1.Invoices                               -- 3. from table `AP1.Invoices`
  INNER JOIN AP1.Terms                            -- 4. `INNER JOIN` to retrieve
                                                  --    data in the first (left)
                                                  --    table (`AP1.Invoices`)
                                                  --    that is also in the
                                                  --    second (right) table
                                                  --    (`AP1.Terms`)
    ON AP1.Invoices.TermsID = AP1.Terms.TermsID   -- 5. `ON` two fields with the
                                                  --    same values/data and the
                                                  --    same name (`TermsID`);
                                                  --    specifying the relation
                                                  --    between tables
                                                  --    `AP1.Invoices` and
                                                  --    `AP1.Terms`
  INNER JOIN AP1.Vendors                          -- 6. `INNER JOIN` to retrieve
                                                  --    data in the second (left)
                                                  --    table (`AP1.Terms`) that
                                                  --    is also in the third
                                                  --    (right) table
                                                  --    (`AP1.Vendors`)
    ON AP1.Vendors.VendorID = AP1.Invoices.VendorID;
                                                  -- 7. `ON` two fields with the
                                                  --    same values/data and the
                                                  --    same name (`VendorID`);
                                                  --    specifying the relation
                                                  --    between tables
                                                  --    `AP1.Vendors` and
                                                  --    `AP1.Invoices`

  /* *****************************************************************************
   https://folvera.commons.gc.cuny.edu/?p=1000 (#1000!)
   **************************************************************************** */
```