CREATING A DATABASE FROM THE CONSOLIDATED SCREENING LIST .CSV FILE FROM TRADE.GOV

I. Collection & Inspection

- Open consolidated.csv
- Select All
- Format by column width
- *OPTIONAL: Download sdn.csv for partial validation

When reviewing the consolidated.csv file, you will need to

- Disregard the schema description on legacy.trade.gov (there is an additional **_id** field)
- Sort columns by **_id** (Column A)
  - These are individual account numbers or identifiers. This will give you give you that intermingles the **ent_num** from the **SDN AND OTHER SANCTIONS LISTS FROM OFAC AND TREASURY** with what appears to be hashed values that serve as individual account numbers for the rest of the entries in this set. Since this is messy, we will separate these values and create a **super_id**.
  - Select All and Sort By **_id** with Order Smallest to Largest.
- *OPTIONAL: Sort by **program** and get the count of **_id**'s that are **SDN**. Then open the **SDN** file and verify the count against the Treasury's data.
  - Make sure to Sort By **_id** before going any further.
- Insert Column into Column A

Before we go any further, let's take a look at the **source** field of our data. Maybe later we can run a query to determine if some departments do not generate these values.

- Scroll down to where the hashed values begin and select the first row. *Control+Shift+Down* to highlight these values, and Sort By **source** with Order Smallest to Largest. Observe the source of these accounts.
  - While we're down here, let's take a look at the highest value of the ordered set. Take note of this for when we're creating the **super_id**. (*later note:* it's currently 7034489)
- Repeat selection for the above set. *Control+Shift+Up* to highlight these values, and Sort By **source** with Order Smallest to Largest and observe the data.

II. Data Organization and Cleansing

Now you are creating the **super_id** which will serve as the new unique account identifier in this database.

- Since you are parsing the original **_id** column and there are only two naming conventions, you will create two methods for generating the **super_id**.
- Begin at the top. Select All and Sort By **_id** with Smallest to Largest.
- In A1, type "=B1+4000000" (4 million). You are preserving the schema of the ent_num to some extent so it might be easier to reference this account if a user is more familiar with that identifier.
    - Double-Click the bottom-right square on the cell to Fill All to the bottom.
    - You will get the *#VALUES!* error on the hashed values. Delete these.
- To reserve room on the **super_id** list for additional accounts imported, we will begin a different naming schema.
- In the first cell (aka. refcell1) with hashed values, type 8000000 (8 million).
- In the cell underneath, type = refcell1+1. Then autofill the rest of the fields by double-clicking the bottom-right square.

Beautiful, our **super_id** has been created. For sanity's sake let's apply a Column Header.

- To retain this hashed data, let's rename the **_id** column to **minor_id**.

Now we can begin cleansing the rest of the data. Remember the source of your data, and the importance of what you are doing. This column represents the various SANCTIONS programs an individual or organization is associated with. Parsing this data is important now to stratify later.

- Bring attention to **programs**
    - In the SDN and other files, this column is named **program** – we would like to follow this convention, but will rename this Header to **program_1**. We can make the same edit to **program** in the SDN list into another table later upon import.
    - **NOTE that each account can belong to multiple programs and that they are separated by a semicolon.

Here is where we use the tools at hand to quickly separate these programs into individual columns for future analysis.

- We're going to use Data > Text to Columns for these.
- Before starting, resize the right-adjacent column and the one to its right for good measure. It is large and makes inserting multiple columns tedious.
- Since we don't know how many columns we need, we're going to use Warnings as our guardrails. Insert 5 columns and test.
    - When run up against Warnings, add more columns.
    - If you are not going incrementally (if you are trying to get a sense of the range), you will need to Undo your work once you get a successful Text to Column and then proceed incrementally.

    - This will determine the current maximum count of programs that are assigned to an individual account. We don’t want to have extra columns in our schema, so for now let’s just determine the current max, then return and go incrementally until we don’t encounter the Warning.
- In Data > Text to Columns, Select *Delimited* and click Next. Then Select Semicolon as the Delimiter. Click Next and leave it as General and Finish. *Again, make sure you confirm you have no additional columns*

Now we need a naming convention for these headers – let’s start with **program_1** and complete the rest, which will continue through **program_8**.

- Sort by **program_8** and observe the **name**. See if there is any information to be gained from observing the subject of 8 international sanctions.
    - You can also add additional Sort fields to see the rest of Top 10.

We want to anticipate that this may not be the upper limit for the number of SANCTIONS programs an account may be affiliated with. For good measure, let’s add two more columns with appropriate headers for a nice round **program_10**.

Observe the **name** column and the format of its data. As we can see, it is not in first name/last name and there are also Aircraft, Entity, and Vessel fields. Let’s begin with Individuals. NOTE: Include an extra field for suffix when adding columns for name fields.

- For an easy way to do this (and really the only way), Select All and Sort By **type** in Order Z to A. We are looking for fields with the Individual value.
- Now we need to add columns to accommodate the new name fields. Add 2 columns right-adjacent to **name**.
- Select only the fields from the **name** field that are also attributed Individual – it may help to temporarily Hide a few columns so that you can see the beginning of the worksheet. *Remember to Unhide.
    - With these, use Data > Text to Columns and Select Comma as the Delimiter. Finish.
- Now we can add the **last_name**, **first_name**, and **suffix_name** to these headers.

We will continue this same process starting with the Aircraft entries. Insert Column and name it **aircraft_name**. From here we can carefully cut and paste our data between adjacent cells. We will create additional **aircraft_name**, **entity_name**, **vessel_name**, and **unknown_account_type_name** using this process.

We are making some meaningful progress now and can already begin to see some of the data come together.

Inspect the **address** column, and observe that this data is not quite clean. For now, we’re going to do what we can to isolate the worst of it and parse the rest.

- Some of the data in here is separated by a semicolon. But on first glance there are many “C/O” references in here, especially between those delimiters. We want to exclude these from our **address** field because we are looking to isolate geographic data as best as possible.

    - We need to create a header for these outliers. Create a column at the end and call it **address_extra**.
- Find and Select all field that contain “C/O” – Select All and apply a Cell Color.
- Escape the selection. Select All, then Sort On **address** with Order set to Cell Color.
- Select these fields within the **address** column

We will need to revisit the **address** field later, but for now let’s keep going.

I am not happy with **address** but for now it’s there. Let’s everything ready for MySQL.

I’m hoping that there will be enough clean data in the **address** field that I can still make some meaningful insights without parsing multiple international regions into new headers (example: *Room 201, 1296 Xuchang Road, Yangpu District, Shanghai, CN; China Communications Building, Block A, Desheng International, No. 85 Deshengmenwai Street, Xicheng District, Beijing, CN*).

We are also going to Data > Text to Column the **citizenships**, **dates_of_birth**, and **nationalities** columns.

- For **citizenships**, Sort By **citizenships** and determine the columns you will need (there are probably not many dual+ citizens). *spoiler: 3 total columns.
- For **dates_of_birth**, Sort By **dates_of_birth** and determine the number of columns you will need. *Spoiler: it’s 9.
- For **nationalities**, Sort By **nationalities** and determine the number of columns you will need. *Spoiler: it’s 4.

Let’s clean some of the values we separated from the **dates_of_birth** Column by renaming the original column **dob_1** and continuing through **dob_9**. Select All and Sort By **dob_1** and Order Oldest to Newest. Observe the data. Some are dates, other are ranges, and others are an estimate (“Circa”).

- To be honest, I’m not quite set on the best way of formatting these custom date ranges from text to DATE format. We’re going to take the entries that say “YEAR to YEAR” and for now store them as a string value in a new **dob_range_string** column.
- Select All and Sort By **dob_1** in Order Smallest to Largest. Select and Cut the fields written in “YYYY to YYYY” format and paste into the **dob_range_string** column.
- Search and Replace “CIRCA ” with “” to isolate the integer
- Now Select All and Sort By dob_1. Select All **dob_#** columns and Format Cells as YYYY-MM-DD.

We will continue the separation and naming conventions for **citizenship_#** and **nationality_#**. The below list should serve as a comprehensive summary of the changes we make to the Source Headers:

Header Columns that were renamed:

- **_id** becomes **minor_id**
- **entity_num** becomes **ent_num**.
- **source** becomes **account_source.**
- **type** becomes **account_type**.
- **programs** is split and becomes **program_1** and continues through **program_10**.
- **addresses** becomes **address_1**. This may need to be parsed into more columns later.

- **name** is split and becomes **last_name**, **first_name**, **suffix_name**, **aircraft_name**, **entity_name**, **vessel_name**, and **unknown_account_type_name**.
- **start_date** and **end_date** become **date_start** and **date_end**, respectively.
- **citizenships** is split and becomes **citizenship_1** and continues through **citizenship_3**.
- **dates_of_birth** is split and becomes **dob_1** and continues through **dob_9**.
  - To account for date ranges, we will also add **dob_range**.
- **nationalities** is split and becomes **nationality_1** and continues through **nationality_3**.
- **ids** becomes **misc_ids**.

Header Columns that were added:

- **super_id**
- **address_extra** (this is the field where we place the "C/O" **address_1** results)

Save your file as "FINAL_cons.csv"

III. Transformation and Visualization

Let's begin writing our query to define our tables. I'll be writing in VS Code and opening within MySQL. To note, some of the Header names in the Source file are reserved words in SQL. This is in part why some of the above changes were made to the dataset. NOTE: You will need to make sure that localhost privileges are set up if you are hosting your MySQL instance locally. (*"SET GLOBAL local_infile=1;"*)

SCHEMA:

```
SET GLOBAL local_infile=1; -- if hosting locally
CREATE DATABASE commerce;
USE commerce;
CREATE TABLE blacklist(
    super_id INT NOT NULL,
    minor_id VARCHAR(500),
    account_source VARCHAR (500),
    ent_num INT,
    account_type VARCHAR(150),
    program_1 VARCHAR(100),
    program_2 VARCHAR(100),
    program_3 VARCHAR(100),
    program_4 VARCHAR(100),
    program_5 VARCHAR(100),
    program_6 VARCHAR(100),
    program_7 VARCHAR(100),
    program_8 VARCHAR(100),
    program_9 VARCHAR(100),
    program_10 VARCHAR(100),
    aircraft_name VARCHAR(100),
    entity_name VARCHAR(250),
    vessel_name VARCHAR(250),
    unknown_account_type_name VARCHAR(500),
    last_name VARCHAR(500),
    first_name VARCHAR(500),
    suffix_name VARCHAR(20),
    title VARCHAR(500),
    address_1 VARCHAR(1000),
    federal_register_notice VARCHAR(800),
    date_start DATE,
    date_end DATE,
    standard_order VARCHAR(10),
    license_requirement VARCHAR(500),
    license_policy VARCHAR(500),
    call_sign VARCHAR(25),
    vessel_type VARCHAR(100),
    gross_tonnage INT,
    gross_registered_tonnage INT,
    vessel_flag VARCHAR(100),
    vessel_owner VARCHAR(100),
    remarks VARCHAR(1001),
    source_list_url VARCHAR(50),
    alt_names VARCHAR(500),
    citizenship_1 VARCHAR(500),
    citizenship_2 VARCHAR(10),
    citizenship_3 VARCHAR(10),
    dob_1 DATE,
    dob_2 DATE,
    dob_3 DATE,
    dob_4 DATE,
    dob_5 DATE,
    dob_6 DATE,
    dob_7 DATE,
    dob_8 DATE,
    dob_9 DATE,
    dob_range_string VARCHAR(255),
    nationality_1 VARCHAR(500),
    nationality_2 VARCHAR(500),
    nationality_3 VARCHAR(500),
    nationality_4 VARCHAR(500),
    places_of_birth VARCHAR(500),
    source_information_url VARCHAR (150),
    misc_ids VARCHAR(1000),
    address_extra VARCHAR(1000),
    PRIMARY KEY (super_id)
);
```

To verify the table was loaded correctly, run `DESC blacklist;` and observe the Column fields.

Now you will need to load your cleansed data file into the table.

Currently, I am running up against warnings with most occurring in the **dob_#** or **gross_registered_tonnage** fields. I believe it is an issue in some *null* non-string fields. I will include the Warnings in a separate file and update after testing Default INT and DATE values.

These instructions assume you have kept the Header fields. If you have deleted them, simply delete the last `IGNORE 1 LINES;` line in the below code.

```
LOAD DATA LOCAL INFILE "source\\FINAL_cons.csv" -- make sure you use \\ in
your file path
INTO TABLE blacklist
COLUMNS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
ESCAPED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES;
```

A separate Warnings file will be uploaded later, but the count is currently at 1024. I am having issues logging these files, the Response will only return “1024 Rows” – I will need to configure MySQL logging and update the Schema, but for now let’s test out database with some simple queries.

Let’s start with a simple `SELECT * FROM blacklist;` - does it work? It should!

Now you can begin testing your database with whichever queries you like; a few samples are provided below.

```
-- number of distinct accounts
SELECT super_id, count(distinct super_id)
FROM blacklist;


-- number of total accounts and verification that super_id generation worked
correctly, counts should be the same as above
SELECT super_id, count(super_id)
FROM blacklist;


-- account_type that are Vessels with their primary Program as Cuba
SELECT * FROM blacklist
WHERE program_1 = 'CUBA'
AND account_type = 'VESSEL';
```