The objective of this exercise is to create a pipeline that ingests, cleans and joins three different files and outputs the final dataframe to a user specified file name of format .csv or .json. No README file is required, however you should write your code in a readable manner and document with comments.

The deliverable is a python script that will be called from the terminal and take an output file path as input, examples on how the script will be called below:

```
python <script.py> output.csv
python <script.py> output.json
```

Note, the output type should be automatically inferred from the file path provided.

Provided files:

employers.csv: contains basic information about our current clients.
quotes.csv: contains information about quoting a new product to our existing clients. The column EmployerId will be the key to join to employers.csv.
product_premiums.txt: This text file contains information about a clients current product mix and the premiums they are paying for each product. This file is not structured in a way that is readable by pandas.

Pipeline steps

Your pipeline should follow these steps:

1. Combine Employers and Quotes data

read the `employers.csv` and `quotes.csv` and merge them on EmployerId as an inner join. Ensure that the result is unique on EmployerId. If there are duplicates, keep the row with the latest quote date.

2. Create a new column named ClientTenure

based on the date the employer became a client and the date ProductX was quoted, determine the client tenure (in years, assuming 365 days a year) at the time of quote. Round to nearest integer.

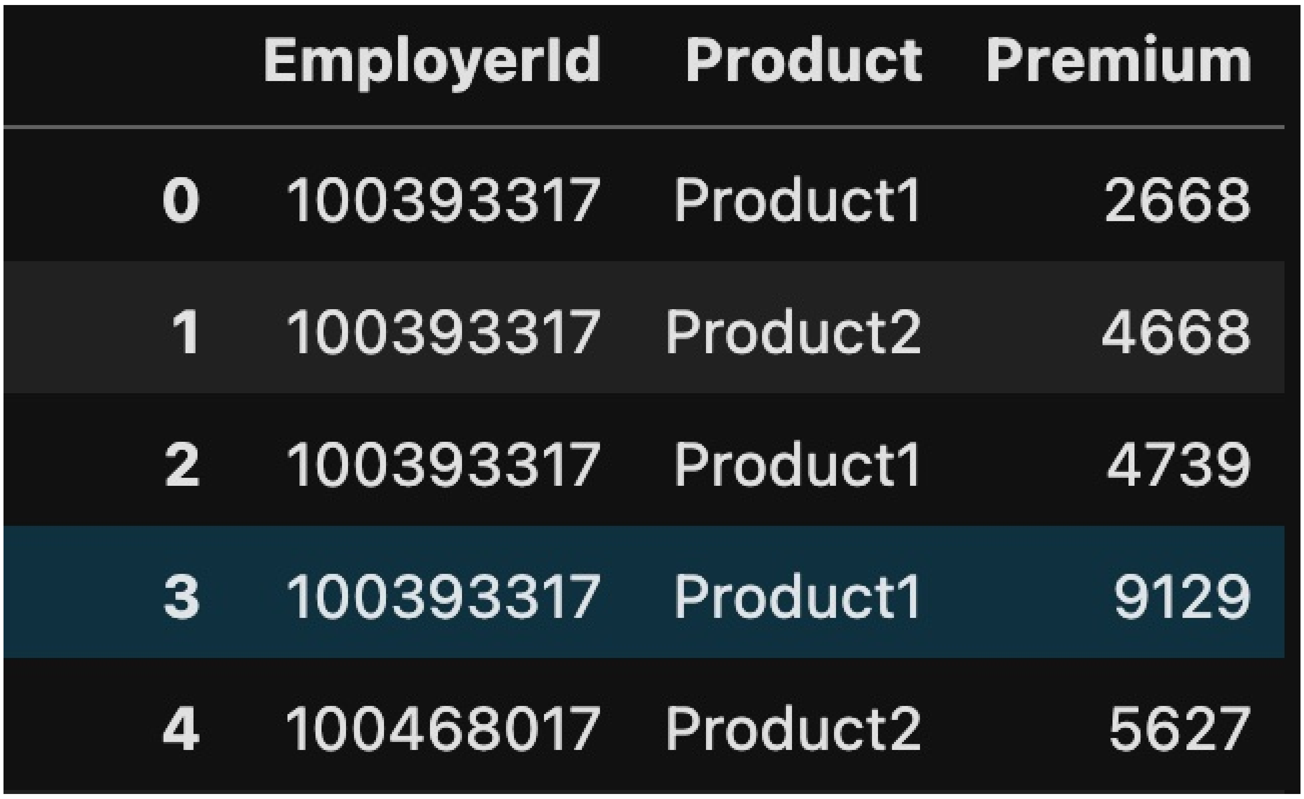
3. Create a column named SizeRange

This will be a categorical column based on the Employees column, which is a count of the number of employees a client has. The resulting values in the new column will be:

```
['0-9 EE', '10-49 EE', '50-99 EE', '100-499 EE', '500-999 EE', '1000-4999
EE', '5000+ EE']
```

4. Read product_premiums.txt

This file contains the products a client currently has, and the annual premium associated with each product. The file structure is not natively readable by pandas, so you must write a parser to extract the information into a dataframe. The file is structured to have a section start with `EmployerId: 00000000` followed by a series of product and premium pairs, such as `Product1: 9233`. Each Id only occurs once. An employer can have duplicate products (one per location) but premiums will be unique. The resulting dataframe should look like this:



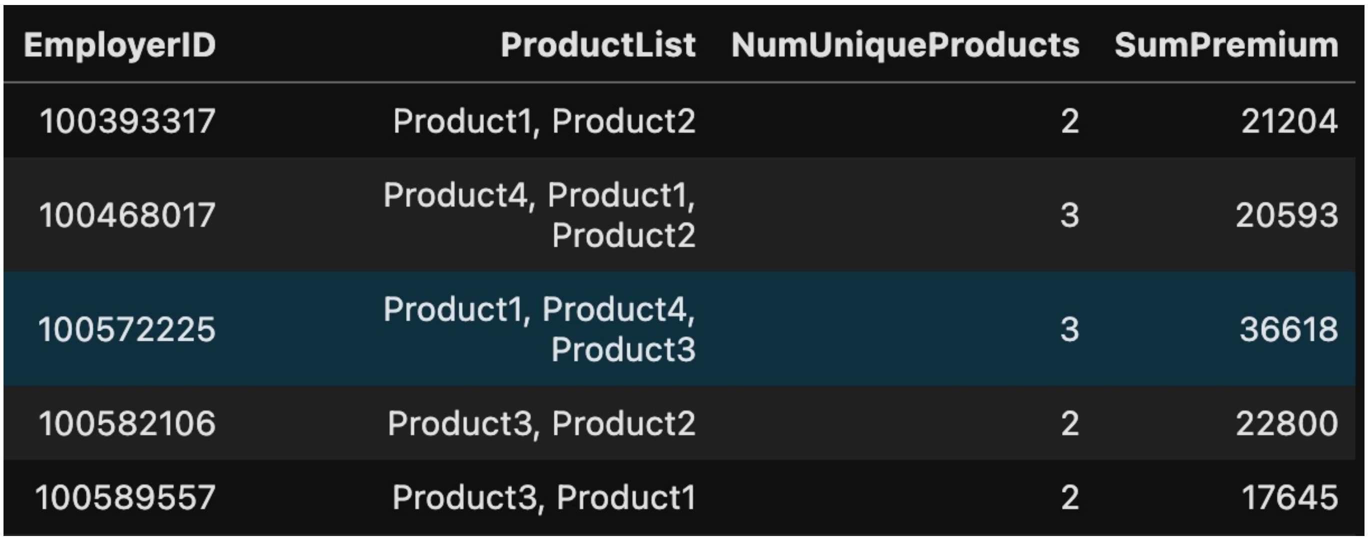
	EmployerId	Product	Premium
0	100393317	Product1	2668
1	100393317	Product2	4668
2	100393317	Product1	4739
3	100393317	Product1	9129
4	100468017	Product2	5627

5. Aggregate the premium data from step 4 to the EmployerId level

Create the following columns:

- `ProductList`: a comma separated string with the distinct products a client has. ex. `'Product4, Product1, Product2'`
- `NumUniqueProducts`: the number of unique products a client has. In the above example it would be 3.
- `SumPremium`: A sum of the premium from all products.

The resulting aggregated dataframe should look like this:



EmployerID	ProductList	NumUniqueProducts	SumPremium
100393317	Product1, Product2	2	21204
100468017	Product4, Product1, Product2	3	20593
100572225	Product1, Product4, Product3	3	36618
100582106	Product3, Product2	2	22800
100589557	Product3, Product1	2	17645

6. Join aggregated premium data to the dataset created in steps 1-3 using pandasql

Write SQL to create an inner join on EmployerId. There will be a case statement, and the columns should be selected as follows:

```
    left.EmployerId,
    left.Employees,
    left.SizeRange,
    left.ClientTenure,
    left.Industry,
    left.ClientStartDate,
    left.QuoteDate,
    left.Offered,
    /* USE CASE STATEMENT to create new column "Sold" as a 1/0 indicator
if product was 'Lost': 0 or 'Sold': 1 */
    right.ProductList as ExistingProducts,
    right.NumUniqueProducts,
    right.SumPremium
```

Note that the aliases are not valid, they just represent which dataframe to pull the columns from. To use pandasql:

```
pip install pandasql

from pandasql import sqldf
pysqldf = lambda q: sqldf(q, globals())

# From https://pypi.org/project/pandasql/

q = """
SELECT
    m.date, m.beef, b.births
FROM meats m
    INNER JOIN births b
        ON m.date = b.date;
```

```
"""
joined = pyqldf(q)
```

7. Write final DataFrame to file output specified by user

user specified file path to be read from command line. A user can specify a .csv file or .json file, but will not provide explicit instruction besides the file extension provided.