Website (https://www.coleridgeinitiative.org)

Ghani, Rayid, Frauke Kreuter, Julia Lane, Adrianne Bradford, Alex Engler, Nicolas Guetta Jeanrenaud, Graham Henke, Daniela Hochfellner, Clayton Hunter, Brian Kim, Avishek Kumar, Jonathan Morgan, Ursula Kaczmarek, Benjamin Feder.

source to be updated when notebook added to GitHub

Dataset Exploration

Table of Contents

JupyterLab contains a dynamic Table of Contents that can be accessed by clicking the last of the six icons on the left-hand sidebar.

Access the Data

In [ ]:
```
# pandas-related imports
import pandas as pd

# database interaction imports
from pyathenajdbc import connect
```

In [ ]:
```
conn = connect(s3_staging_dir = 's3://usda-iri-2019-queryresults/',
               region_name = 'us-gov-west-1',
               LogLevel = '0',
               workgroup = 'workgroup-iri_usda')
```

Analysis: Using Python and SQL

Do WIC households purchase similar foods compared to households that are WIC-eligible but not participants in 2017?

To find the answers to these questions, we will need to combine the demographics and trips data with another available dataset. Our process will work as follows:

- Explore the available data tables
- Check out Demographics and Trips tables in the context of WIC eligibility
- Join datasets
- Join with third dataset
- Answer questions

Necessary data:

- `iri_usda.demo_all` : individual household demographics data
- `iri_usda.trip_all` : individual purchase data for every item that every household purchased
- `iri_usda.pd_pos_all` : comprehensive product data

Demographics Table

Similar to the first data exploration notebook, we will start sifting through `iri_usda.demo_all` to help answer our guiding question.

Let's explore a few specific questions to better understand our data in 2017:

- How many households are already enrolled in the WIC program and have sufficient purchase data?
- How can we determine if a household is WIC-eligible based on the provided data?
- How many households with sufficient purchase data are WIC-eligible but aren't enrolled?

In [ ]:
```
# count of households with sufficient purchase data in WIC program in 20
17
query = '''
SELECT count(*) as wic_households
FROM iri_usda.demo_all
WHERE wic_june = 1 and year = '2017' and projection61k > 0;
'''

pd.read_sql(query, conn)
```

If you search for the requirements for WIC eligibility, you will see that there are four categories:

1. Cateogrical
2. Residential
3. Income
4. Nutrition Risk

Given the data and general knowledge of the households, we cannot answer as to whether or not they are deemed a nutritional risk. However, we can find households that with young children and we have household size to income data. Thus, we will do our best to estimate which households may be WIC-eligible. To do so, we will need to examine three variables in the `iri_usda.demo_all` dataset: `ac`, `hhsize`, and `hhinc`. We are using the available 2017-18 income guidelines for WIC households as per WIC Policy Memo #2017-3.

In [ ]:
```
# number of non-wic but income-eligible participants with sufficient pur
chase data in 2017
query = """
SELECT count(*) as wic_eligible
FROM iri_usda.demo_all
WHERE ac in (1,4,5,7) and
    (
    (hhinc <= 5 and hhsize = 1) or (hhinc = 6 and hhsize < 3) or (hhinc
 = 7 and hhsize < 4) or
    (hhinc = 8 and hhsize < 5) or (hhinc = 9 and hhsize < 6) or (hhinc =
10 and hhsize < 8) or
    (hhinc = 11 and hhsize = 8)
    )
    and wic_june != 1 and year = '2017' and projection61k > 0;
"""

df = pd.read_sql(query, conn)

df.head()
```

In the context of the guiding question, we only need to focus on households that are either enrolled in the WIC program in 2017 or were wic-eligible to the best of our knowledge in 2017. To make things easier and quicker, we already created a table `project_q2_cohort` in the `iri_usda_2019_db` database that contains demographic data for just this select group of households in 2017 with sufficient purchasing data. The code used to create this table is shown below.

> Note: We will not be applying the weights to individual product quantities here because the weights were not constructed on the product level.

```
create table iri_usda_2019_db.project_q2_cohort
    with(
    format = 'Parquet',
    parquet_compression = 'SNAPPY'
    )
    as
    SELECT *
    FROM iri_usda.demo_all
    WHERE ((ac in (1,4,5,7) and
    (
    (hhinc <= 5 and hhsize = 1) or (hhinc <= 6 and hhsize = 2) or (hhinc <=
 7 and hhsize = 3) or
    (hhinc <= 8 and hhsize = 4) or (hhinc <= 9 and hhsize = 5) or (hhinc <=
 10 and hhsize = 6) or
        (hhinc <= 11 and (hhsize = 8 or hhsize = 7))
    )
    and wic_june != 1) or wic_june = 1) and year = '2017' and projection61k
 > 0;
```

Revisiting Trip Data and Join to Cohort

Since we've already gone through the trip data once before, we will start with finding close to what we need: *How can we find the most popular goods in 2017?*

Afterwards, we can write code outputting the most popular goods in 2017 for WIC households with sufficient purchase data.

In [ ]:
```
# get entries of 10 most popular goods
query = """
SELECT upc, sum(quantity) as total
FROM iri_usda.trip_all
WHERE year = '2017'
GROUP BY upc
ORDER BY total desc
LIMIT 10
"""
# print results
df = pd.read_sql(query, conn)

print(df.head())
```

In [ ]:
```
# joining data to get 10 most popular upc codes for WIC households with
 sufficient purchasing data in 2017

query = """
SELECT trip.upc, sum(trip.quantity) as total
FROM iri_usda_2019_db.project_q2_cohort demo
LEFT JOIN iri_usda.trip_all trip
ON trip.panid = demo.panid
WHERE demo.wic_june = 1 and trip.year = '2017'
GROUP BY trip.upc
ORDER BY total desc
LIMIT 10;
"""
# display result
pd.read_sql(query, conn)
```

Checkpoint 1: Popular Goods

Given the code above, how can you find the upc codes corresponding to the 10 most popular goods for WIC-eligible households with sufficient purchase data? **Discuss with your group.**

In [ ]:
```
# get entries of 10 most popular upc codes for wic-eligible households w
ith sufficient purchasing data in 2017
```

Product Descriptions

How can we find the descriptions of these most popular goods?

We will need to join the purchasing dataset with `iri_usda.pd_pos_all`, which contains the product descriptions for each `upc` provided. Having the descriptions of these goods for our two cohorts will give us a better idea of what each group purchases the most, rather than just referring to the goods by their upc codes. The variable in `pd_pos_all` that contains the product descriptions corresponding to the upc codes is `upcdesc`.

First, we will make sure we can join `trip_all` to `pd_pos_all` before getting the corresponding descriptions of the 100 most popular products for our two cohorts.

> Note: We created two tables, `pop_upc_wic` and `pop_upc_eligible` in the `iri_usda_2019_db` database for you that contain the 100 most popular products in 2017 for wic-participant and wic-eligible households, respectively, with sufficient purchasing data.

In [ ]:
```
# join trip_all and pd_pos_all
query = '''
select trip.upc, product.upcdesc
from iri_usda.trip_all trip
left join iri_usda.pd_pos_all product
on product.upc = trip.upc
limit 10
'''

pd.read_sql(query, conn)
```

In [ ]:
```
# explore pop_upc_wic
qry = '''
select *
from iri_usda_2019_db.pop_upc_wic
limit 10
'''

pd.read_sql(qry, conn)
```

In [ ]:
```
# explore pop_upc_eligible
qry = '''
select *
from iri_usda_2019_db.pop_upc_eligible
limit 10
'''

pd.read_sql(qry, conn)
```

In [ ]:
```
# find names for 100 most popular goods for wic households with sufficie
nt purchase data in 2017
query = '''
select distinct wic.upc, wic.total, product.upcdesc
from iri_usda_2019_db.pop_upc_wic wic
left join iri_usda.pd_pos_all product
on wic.upc = product.upc
where product.upcdesc != ''
'''

pd.read_sql(query, conn)
```

Checkpoint 2: Reality Check

Recreate the same table for non-WIC but WIC-eligible households. Do the most popular goods vary between the two groups? Do the results make sense to you? Why or why not? **Discuss with your group.**

In [ ]:
```
# find names for 100 most popular goods for WIC-eligible households with
sufficient purchase data in 2017
query = '''
INSERT QUERY
'''

pd.read_sql(query, conn)
```

Moving Forward

Based off of the guiding questions in these notebooks, we will extend these analyses a bit further in the sample project notebooks.