In [4]:
```
import pandas as pd
import numpy as np


df_community = pd.read_csv('/Users/cinema/Desktop/nunavut_communities_da

df_price_list = pd.read_csv('/Users/cinema/Desktop/fresh_produce_dataset

df_community.head
df_price_list.head
```

Out[4]:
```
<bound method NDFrame.head of    product_type_beef  price_per_one_kg_
beef location_beef  \
0              bacon                  7.25       alberta
1        blade roast                 18.49       alberta
2    broiler chicken                  7.41       alberta
3        ground beef                 12.50       alberta
4         pork chops                 12.39        quebec
5    prime rib roast                 37.09       alberta
6        round steak                 18.39       alberta
7      sirloin steak                 24.45       alberta
8       stewing beef                 18.31       alberta
9            wieners                  4.47             0
10                 0                  0.00             0
11                 0                  0.00             0
12                 0                  0.00             0
13                 0                  0.00             0
14                 0                  0.00             0
15                 0                  0.00             0
16                 0                  0.00             0
```

In [11]:
```
import pandas as pd
import numpy as np


df_community = pd.read_csv('/Users/cinema/Desktop/nunavut_communities_da

df_price_list = pd.read_csv('/Users/cinema/Desktop/fresh_produce_dataset

#get produce values from our fresh_produce_dataset.
#beef
product_type_beef = df_price_list['product_type_beef']
location_beef = df_price_list['location_beef']
price_per_one_kg_beef = df_price_list['price_per_one_kg_beef']
produce_lifetime_days_beef = df_price_list['produce_lifetime_days_beef']
```

```
#dairy
product_type_dairy = df_price_list['product_type_dairy']
location_dairy = df_price_list['location_dairy']
price_per_one_kg_dairy = df_price_list['price_per_litre_kg']
produce_lifetime_days_dairy = df_price_list['produce_lifetime_days_dairy

#fruit
product_type_fruit = df_price_list['product_type_fruit']
location_fruit = df_price_list['location_fruit']
price_per_one_kg_fruit = df_price_list['price_per_kg_fruit']
produce_lifetime_days_fruit = df_price_list['produce_lifetime_days_fruit


#grain
product_type_grain = df_price_list['product_type_grain']
location_grain = df_price_list['location_grain']
price_per_bushel_grain = df_price_list['price_per_bushel_grain']
produce_lifetime_days_grain = df_price_list['produce_lifetime_days_grain

#vegetable
product_type_vegetables = df_price_list['product_type_vegetables']
location_vegetables = df_price_list['location_vegetables']
price_per_one_kg_vegetables = df_price_list['price_per_kg_vegetables']
produce_lifetime_days_vegetables = df_price_list['produce_lifetime_days_v
```

Out[11]:
```
<bound method NDFrame.head of 0           artichoke
1           asparagus
2              garlic
3         green beans
4               beets
5         bell pepper
6            broccoli
7     brussel sprouts
8             cabbage
9             carrots
10        cauliflower
11             celery
12            cucmber
13               kale
14            lettuce
15              onion
16            parsnip
17             potato
18            spinach
19             tomato
20                  0
21                  0
Name: product_type_vegetables, dtype: object>
```

In [12]:
```
def getProducePriceMinOverhead(collectProduce, total):
    return collectProduce - total

#method for collecting a specific row in a column for a specific produce
def collectProduce(produce, produce_shelflife, produce_price_per_kg):
    df_produce = produce
    df_shelflife = float(produce_shelflife)
    df_produce_price = float(produce_price_per_kg)
    price_per_60000_kg = df_produce_price * 60000
    return price_per_60000_kg
```

In [13]:
```
#overhead costs
personnel_cost = df_community['personnel_cost']
storage_cost = df_community['storage_cost']
air_freight_cost = df_community['freightfare_per_region_based_off_kg_of_
```

In [14]:
```
total_overhead = 679279.16
total_kg = 60000
```

In [15]:
```
#produce cost for 60,000 KG worth of different produce.
#beef

bacon = collectProduce(product_type_beef[0], produce_lifetime_days_beef[
blade_roast = collectProduce(product_type_beef[1], produce_lifetime_days_
broiler_chicken = collectProduce(product_type_beef[2], produce_lifetime_
ground_beef = collectProduce(product_type_beef[3], produce_lifetime_days_
pork_chops = collectProduce(product_type_beef[4], produce_lifetime_days_
prime_rib_roast = collectProduce(product_type_beef[5], produce_lifetime_
round_steak = collectProduce(product_type_beef[6], produce_lifetime_days_
sirloin_steak = collectProduce(product_type_beef[7], produce_lifetime_day
stewing_beef = collectProduce(product_type_beef[8], produce_lifetime_days
wieners = collectProduce(product_type_beef[9], produce_lifetime_days_bee
```

In [16]:
```
#group all current retail prices of beef types into one list
beef = [bacon,blade_roast,broiler_chicken,ground_beef,pork_chops,prime_r
```

In [17]:
```
#total overhead cost subtracted from the total cost of 60,000 kg of beef
bacon_tot_min_overhead_cost = getProducePriceMinOverhead(bacon, total_ove
blade_roast_tot_min_overhead_cost = getProducePriceMinOverhead(blade_roas
broiler_chicken_tot_min_overhead_cost = getProducePriceMinOverhead(broile
ground_beef_tot_min_overhead_cost = getProducePriceMinOverhead(ground_bee
pork_chops_tot_min_overhead_cost = getProducePriceMinOverhead(pork_chops
prime_rib_tot_min_overhead_cost = getProducePriceMinOverhead(prime_rib_r
round_steak_tot_min_overhead_cost = getProducePriceMinOverhead(round_stea
sirloin_steak_tot_min_overhead_cost = getProducePriceMinOverhead(sirloin_
stewing_beef_tot_min_overhead_cost = getProducePriceMinOverhead(stewing_b
wieners_tot_min_overhead_cost = getProducePriceMinOverhead(wieners, total

produce_60k_kg_min_tot_overhead_costs_list = [bacon_tot_min_overhead_cost
```

In [18]:
```
#adjusted price is TOC from above divided by total produce size in KG to
adj_bacon = getProducePriceMinOverhead(bacon, total_overhead) / total_kg
adj_blade_roast = getProducePriceMinOverhead(blade_roast, total_overhead
adj_broiler_chicken = getProducePriceMinOverhead(broiler_chicken, total_o
adj_ground_beef = getProducePriceMinOverhead(ground_beef, total_overhead
adj_pork_chops = getProducePriceMinOverhead(pork_chops, total_overhead) /
adj_prime_rib_roast = getProducePriceMinOverhead(prime_rib_roast, total_o
adj_round_steak = getProducePriceMinOverhead(round_steak, total_overhead
adj_sirloin_steak = getProducePriceMinOverhead(sirloin_steak, total_overh
adj_stewing_beef = getProducePriceMinOverhead(stewing_beef, total_overhea
adj_wieners = getProducePriceMinOverhead(wieners, total_overhead) / total

new_price_per_kg_of_produce = [adj_bacon,adj_blade_roast,adj_broiler_chic
```

In [19]:
```
 = collectProduce(product_type_dairy[0],produce_lifetime_days_dairy[0],pr
ctProduce(product_type_dairy[1],produce_lifetime_days_dairy[1],price_per_

ead cost subtracted from the total cost of 60,000 kg of dairy
min_overhead = getProducePriceMinOverhead(chicken_eggs, total_kg)
rhead = getProducePriceMinOverhead(milk, total_kg)

ice is TOC from above divided by total produce size in KG to determine co
st per unit of 60,000 kg of produce after subtracting overhead.
egg = chicken_egg_min_overhead / total_kg
ilk_min_overhead / total_kg

ice_list = [adj_chicken_egg, adj_milk]
```

In [20]:
```
#fruit
apple = collectProduce(product_type_fruit[0],produce_lifetime_days_fruit
apricot = collectProduce(product_type_fruit[1],produce_lifetime_days_fru
avocado = collectProduce(product_type_fruit[2],produce_lifetime_days_fru
banana = collectProduce(product_type_fruit[3],produce_lifetime_days_frui
blueberry = collectProduce(product_type_fruit[4],produce_lifetime_days_f
cherries = collectProduce(product_type_fruit[5],produce_lifetime_days_fr
cranberry = collectProduce(product_type_fruit[6],produce_lifetime_days_f
grapefruit = collectProduce(product_type_fruit[7],produce_lifetime_days_
grapes = collectProduce(product_type_fruit[8],produce_lifetime_days_frui
guava = collectProduce(product_type_fruit[9],produce_lifetime_days_fruit
orange = collectProduce(product_type_fruit[10],produce_lifetime_days_fru
papaya = collectProduce(product_type_fruit[11],produce_lifetime_days_fru
peach = collectProduce(product_type_fruit[12],produce_lifetime_days_frui
pear = collectProduce(product_type_fruit[13],produce_lifetime_days_fruit
```

In [21]:
```
#total overhead cost subtracted from the total cost of 60,000 kg of frui
apple_min_overhead = getProducePriceMinOverhead(apple, total_kg)
apricot_min_overhead = getProducePriceMinOverhead(apricot, total_kg)
avocado_min_overhead = getProducePriceMinOverhead(avocado, total_kg)
banana_min_overhead = getProducePriceMinOverhead(banana, total_kg)
blueberry_min_overhead = getProducePriceMinOverhead(blueberry, total_kg)
cherries_min_overhead = getProducePriceMinOverhead(cherries, total_kg)
cranberry_min_overhead = getProducePriceMinOverhead(cranberry, total_kg)
grapefruit_min_overhead = getProducePriceMinOverhead(grapefruit, total_k
grapes_min_overhead = getProducePriceMinOverhead(grapes, total_kg)
guava_min_overhead = getProducePriceMinOverhead(guava, total_kg)
orange_min_overhead = getProducePriceMinOverhead(orange, total_kg)
papaya_min_overhead = getProducePriceMinOverhead(papaya, total_kg)
peach_min_overhead = getProducePriceMinOverhead(peach, total_kg)
pear_min_overhead = getProducePriceMinOverhead(pear, total_kg)
```

In [22]:
```
#adjusted cost per unit of 60,000 kg of produce after subtracting overhea

adj_apple = apple_min_overhead / total_kg
adj_apricot = apricot_min_overhead / total_kg
adj_avocado = avocado_min_overhead / total_kg
adj_banana = banana_min_overhead / total_kg
adj_blueberry = blueberry_min_overhead / total_kg
adj_cherries = cherries_min_overhead / total_kg
adj_cranberry = cranberry_min_overhead / total_kg
adj_grapefruit = grapefruit_min_overhead / total_kg
adj_grapes = grapes_min_overhead / total_kg
adj_guava = guava_min_overhead / total_kg
adj_orange = orange_min_overhead / total_kg
adj_papaya = papaya_min_overhead / total_kg
adj_peach = peach_min_overhead / total_kg
adj_pear = pear_min_overhead / total_kg
```

In [23]:
```
#grain
barley = collectProduce(product_type_grain[0],produce_lifetime_days_grai
canola = collectProduce(product_type_grain[1],produce_lifetime_days_grai
corn = collectProduce(product_type_grain[2],produce_lifetime_days_grain[
flaxseed = collectProduce(product_type_grain[3],produce_lifetime_days_gra
oats = collectProduce(product_type_grain[4],produce_lifetime_days_grain[
peas = collectProduce(product_type_grain[5],produce_lifetime_days_grain[
soybean = collectProduce(product_type_grain[6],produce_lifetime_days_gra
wheat = collectProduce(product_type_grain[7],produce_lifetime_days_grain
```

In [24]:
```
#total overhead cost subtracted from the total cost of 60,000 kg of grai
barley_min_overhead = getProducePriceMinOverhead(barley, total_kg)
canola_min_overhead = getProducePriceMinOverhead(canola, total_kg)
corn_min_overhead = getProducePriceMinOverhead(corn, total_kg)
flaxseed_min_overhead = getProducePriceMinOverhead(flaxseed, total_kg)
oats_min_overhead = getProducePriceMinOverhead(oats, total_kg)
peas_min_overhead = getProducePriceMinOverhead(peas, total_kg)
soybean_min_overhead = getProducePriceMinOverhead(soybean, total_kg)
wheat_min_overhead = getProducePriceMinOverhead(wheat, total_kg)
```

In [25]:
```
#adjusted cost per unit of 60,000 kg of produce after subtracting overhe

barley = barley_min_overhead / total_kg
canola = canola_min_overhead / total_kg
corn = corn_min_overhead / total_kg
flaxseed = flaxseed_min_overhead / total_kg
oats = oats_min_overhead / total_kg
peas = peas_min_overhead / total_kg
soybean = soybean_min_overhead / total_kg
wheat = wheat_min_overhead / total_kg
```

In [26]:
```
#vegetables
artichoke = collectProduce(product_type_vegetables[0],produce_lifetime_da
asparagus = collectProduce(product_type_vegetables[1],produce_lifetime_da
garlic = collectProduce(product_type_vegetables[2],produce_lifetime_days_
green_beans = collectProduce(product_type_vegetables[3],produce_lifetime_
beets = collectProduce(product_type_vegetables[4],produce_lifetime_days_v
bell_peppers = collectProduce(product_type_vegetables[5],produce_lifetime
broccoli = collectProduce(product_type_vegetables[6],produce_lifetime_day
brussel_sprouts = collectProduce(product_type_vegetables[7],produce_life
cabbage = collectProduce(product_type_vegetables[8],produce_lifetime_days
carrots = collectProduce(product_type_vegetables[9],produce_lifetime_days
cauliflower = collectProduce(product_type_vegetables[10],produce_lifetime
celery = collectProduce(product_type_vegetables[11],produce_lifetime_days
cucumber = collectProduce(product_type_vegetables[12],produce_lifetime_da
kale = collectProduce(product_type_vegetables[13],produce_lifetime_days_v
lettuce = collectProduce(product_type_vegetables[14],produce_lifetime_day
onion = collectProduce(product_type_vegetables[15],produce_lifetime_days_
parsnip = collectProduce(product_type_vegetables[16],produce_lifetime_day
potato = collectProduce(product_type_vegetables[17],produce_lifetime_days
spinach = collectProduce(product_type_vegetables[18],produce_lifetime_day
tomato = collectProduce(product_type_vegetables[19],produce_lifetime_days
```

In [27]:
```
#total overhead cost subtracted from the total cost of 60,000 kg of vege
artichoke_min_overhead = getProducePriceMinOverhead(wheat, total_kg)
asparagus_min_overhead = getProducePriceMinOverhead(asparagus, total_kg)
garlic_min_overhead = getProducePriceMinOverhead(garlic, total_kg)
green_beans_min_overhead = getProducePriceMinOverhead(green_beans, total_
beets_min_overhead = getProducePriceMinOverhead(beets, total_kg)
bell_peppers_min_overhead = getProducePriceMinOverhead(bell_peppers, tota
broccoli_min_overhead = getProducePriceMinOverhead(broccoli, total_kg)
brussel_sprouts_min_overhead = getProducePriceMinOverhead(brussel_sprouts
cabbage_min_overhead = getProducePriceMinOverhead(cabbage, total_kg)
carrots_min_overhead = getProducePriceMinOverhead(carrots, total_kg)
cauliflower_min_overhead = getProducePriceMinOverhead(cauliflower, total_
celery_min_overhead = getProducePriceMinOverhead(celery, total_kg)
cucumber_min_overhead = getProducePriceMinOverhead(cucumber, total_kg)
kale_min_overhead = getProducePriceMinOverhead(kale, total_kg)
lettuce_min_overhead = getProducePriceMinOverhead(lettuce, total_kg)
onion_min_overhead = getProducePriceMinOverhead(onion, total_kg)
parsnip_min_overhead = getProducePriceMinOverhead(parsnip, total_kg)
potato_min_overhead = getProducePriceMinOverhead(potato, total_kg)
spinach_min_overhead = getProducePriceMinOverhead(spinach, total_kg)
tomato_min_overhead = getProducePriceMinOverhead(tomato, total_kg)
```

In [28]:
```
#adjusted cost per unit of 60,000 kg of produce after subtracting overhea
adj_artichoke = artichoke_min_overhead / total_kg
adj_asparagus = asparagus_min_overhead / total_kg
adj_garlic = garlic_min_overhead / total_kg
adj_green_beans = green_beans_min_overhead / total_kg
adj_beets = beets_min_overhead / total_kg
adj_bell_peppers = bell_peppers_min_overhead / total_kg
adj_broccoli = broccoli_min_overhead / total_kg
adj_brussel_sprouts = brussel_sprouts_min_overhead / total_kg
adj_cabbage = cabbage_min_overhead / total_kg
adj_carrots = carrots_min_overhead / total_kg
adj_cauliflower = cauliflower_min_overhead / total_kg
adj_celery = celery_min_overhead / total_kg
adj_cucumber = cucumber_min_overhead / total_kg
adj_kale = kale_min_overhead / total_kg
adj_lettuce = lettuce_min_overhead / total_kg
adj_onion = onion_min_overhead / total_kg
adj_parsnip = parsnip_min_overhead / total_kg
adj_potato = potato_min_overhead / total_kg
adj_spinach = spinach_min_overhead / total_kg
adj_tomato = tomato_min_overhead / total_kg
```

In [29]:
```
#note: personnel get paid 1K CAD and 2.6 kg of free produce of their cho
```

In [ ]:
```

```