

Practical Exam: Grocery Store Sales

FoodYum is a grocery store chain that is based in the United States.

Food Yum sells items such as produce, meat, dairy, baked goods, snacks, and other household food staples.

As food costs rise, FoodYum wants to make sure it keeps stocking products in all categories that cover a range of prices to ensure they have stock for a broad range of customers.

Data

The data is available in the table products.

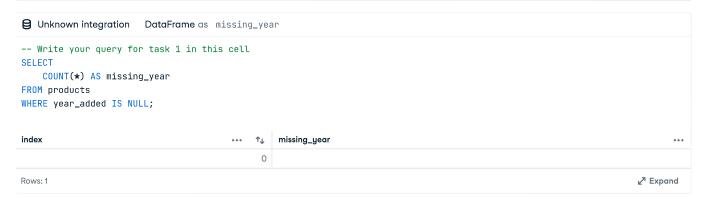
The dataset contains records of customers for their last full year of the loyalty program.

Column Name	Criteria				
product_id	Nominal. The unique identifier of the product. Missing values are not possible due to the database structure.				
product_type	Nominal. The product category type of the product, one of 5 values (Produce, Meat, Dairy, Bakery, Snacks). Missing values should be replaced with "Unknown".				
brand	Nominal. The brand of the product. One of 7 possible values. Missing values should be replaced with "Unknown".				
weight	Continuous. The weight of the product in grams. This can be any positive value, rounded to 2 decimal places. Missing values should be replaced with the overall median weight.				
price	Continuous. The price the product is sold at, in US dollars. This can be any positive value, rounded to 2 decimal places. Missing values should be replaced with the overall median price.				
average_units_sold	Discrete. The average number of units sold each month. This can be any positive integer value. Missing values should be replaced with 0.				
year_added	Nominal. The year the product was first added to FoodYum stock. Missing values should be replaced with 2022.				
stock_location	Nominal. The location that stock originates. This can be one of four warehouse locations, A, B, C or D Missing values should be replaced with "Unknown".				

Task 1

In 2022 there was a bug in the product system. For some products that were added in that year, the year_added value was not set in the data. As the year the product was added may have an impact on the price of the product, this is important information to have.

Write a query to determine how many products have the | year_added | value missing. Your output should be a single column, | missing_year |, with a single row giving the number of missing values.



Task 2

Given what you know about the year added data, you need to make sure all of the data is clean before you start your analysis. The table below shows what the data should look like.

Write a query to ensure the product data matches the description provided. Do not update the original table.

Column Name	Nominal. The unique identifier of the product. Missing values are not possible due to the database structure.					
product_id						
product_type	Nominal. The product category type of the product, one of 5 values (Produce, Meat, Dairy, Bakery, Snacks). Missing values should be replaced with "Unknown".					
brand	Nominal. The brand of the product. One of 7 possible values. Missing values should be replaced with "Unknown".					
weight	Continuous. The weight of the product in grams. This can be any positive value, rounded to 2 decimal places. Missing values should be replaced with the overall median weight.					
price	Continuous. The price the product is sold at, in US dollars. This can be any positive value, rounded to 2 decimal places. Missing values should be replaced with the overall median price.					
average_units_sold	Discrete. The average number of units sold each month. This can be any positive integer value. Missing values should be replaced with 0.					
year_added	Nominal. The year the product was first added to FoodYum stock. Missing values should be replaced with last year (2022).					
stock_location	Nominal. The location that stock originates. This can be one of four warehouse locations, A, B, C or D Missing values should be replaced with "Unknown".					

```
Unknown integration DataFrame as c
-- Write your query for task 2 in this cell
WITH ranked_weights AS (
   SELECT
       weight,
       ROW_NUMBER() OVER (ORDER BY
              ELSE CAST(weight AS NUMERIC)
           END
       ) AS rn,
       COUNT(*) OVER() AS cnt
   FROM products
   WHERE weight IS NOT NULL
),
median_weight AS (
   SELECT ROUND(AVG(CAST(weight AS NUMERIC)), 2) AS median_value
   FROM ranked_weights
   WHERE rn IN ((cnt + 1) / 2, (cnt + 2) / 2)
),
median_price AS (
   SELECT ROUND(AVG(price::NUMERIC), 2) AS median_value
   FROM (
       SELECT price, ROW_NUMBER() OVER (ORDER BY price::NUMERIC) AS rn, COUNT(*) OVER() AS cnt
       FROM products WHERE price IS NOT NULL
   WHERE rn IN ((cnt + 1) / 2, (cnt + 2) / 2)
)
SELECT
   product_id,
   COALESCE(NULLIF(product_type, ''), 'Unknown') AS product_type,
   COALESCE(NULLIF(brand, '-'), 'Unknown') AS brand,
   RUIIND (
       COALESCE(
              WHEN weight \sim E'^\\d+\\.?\\d*\\s*grams$' THEN CAST(REPLACE(weight, ' grams', '') AS NUMERIC)
              ELSE CAST(weight AS NUMERIC)
           END,
           (SELECT median_value FROM median_weight)
       ), 2
   ) AS weight,
   ROUND(
       COALESCE(price::NUMERIC, (SELECT median_value FROM median_price)), 2
   ) AS price,
   COALESCE(average_units_sold::INTEGER, 0) AS average_units_sold,
   COALESCE(year_added::INTEGER, 2022) AS year_added,
   COALESCE(NULLIF(UPPER(stock_location), ''), 'Unknown') AS stock_location
FROM products;
```

↑↓	p ••• ↑↓	prod ••• ↑↓	brand ··· ↑↓	••• ↑↓	••• ↑↓	average_units ◆◆◆	y ••• ↑↓	stock_lo ◆◆◆
0	1	Bakery	TopBrand	602.61	11	15	2022	С
1	2	Produce	SilverLake	478.26	8.08	22	2022	С
2	3	Produce	TastyTreat	532.38	6.16	21	2018	В
3	4	Bakery	StandardYums	453.43	7.26	21	2021	D
4	5	Produce	GoldTree	588.63	7.88	21	2020	Α
5	6	Meat	TopBrand	612.06	16.2	24	2017	Α
6	7	Produce	GoldTree	320.49	8.01	21	2019	В
7	8	Meat	SilverLake	535.19	15.77	28	2021	Α
8	9	Meat	StandardYums	375.07	11.57	30	2020	Α
9	10	Meat	TastyTreat	506.34	13.94	27	2018	С
10	11	Dairy	StandardYums	345.07	9.26	26	2020	В
11	12	Bakery	StandardYums	345.58	6.87	21	2022	D
12	13	Snacks	SmoothTaste	512.54	8.65	19	2016	Α
13	14	Meat	StandardYums	395.76	11.92	30	2019	Α
14	15	Produce	SilverLake	324.92	7.94	23	2021	D
15	16	Dairy	SmoothTaste	446.76	10.79	23	2017	D

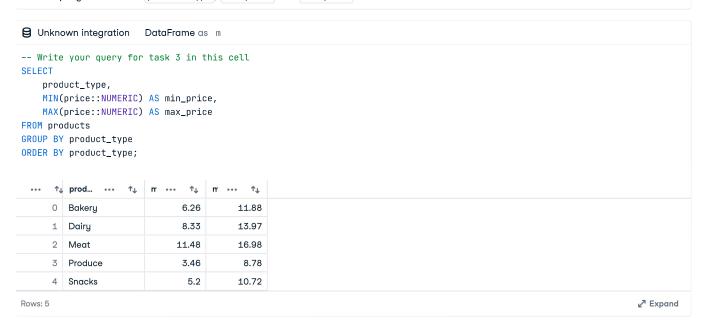
Rows: 1,700

∠³ Expand

Task 3

To find out how the range varies for each product type, your manager has asked you to determine the minimum and maximum values for each product type.

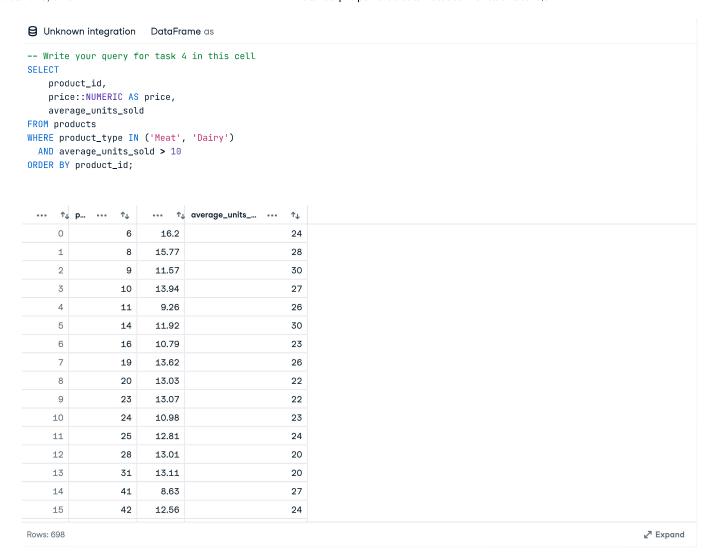
Write a query to return the product_type, min_price and max_price columns.



Task 4

The team want to look in more detail at meat and dairy products where the average units sold was greater than ten.

Write a query to return the product_id, price and average_units_sold of the rows of interest to the team.



FORMATTING AND NAMING CHECK

Use the code block below to check that your outputs are correctly named and formatted before you submit your project.

This code checks whether you have met our automarking requirements: that the specified DataFrames exist and contain the required columns. It then prints a table showing $\[\]$ for each column that exists and $\]$ for any that are missing, or if the DataFrame itself isn't available.

If a DataFrame or a column in a DataFrame doesn't exist, carefully check your code again.

IMPORTANT: even if your code passes the check below, this does not mean that your entire submission is correct. This is a check for naming and formatting only.

```
import pandas as pd
def check_columns(output_df, output_df_name, required_columns):
    results = []
    for col in required_columns:
        exists = col in output_df.columns
        results.append({'Dataset': output_df_name, 'Column': col, 'Exists': '♥' if exists else 'X'})
    return results
def safe_check(output_df_name, required_columns):
    results = []
    if output_df_name in globals():
        obj = globals()[output_df_name]
        if isinstance(obj, pd.DataFrame):
           results.extend(check_columns(obj, output_df_name, required_columns))
        elif isinstance(obj, str) and ("SELECT" in obj.upper() or "FROM" in obj.upper()):
            results.append({'Dataset': output_df_name, 'Column': '-', 'Exists': 'I SQL query string'})
        else:
            results.append({'Dataset': output_df_name, 'Column': '-', 'Exists': 'X Not a DataFrame or query'})
    else:
       results.append({'Dataset': output_df_name, 'Column': '-', 'Exists': 'X Variable not defined'})
    return results
requirements = {
    'missing_year': ['missing_year'],
    'clean_data': ['product_id', 'product_type', 'brand', 'weight', 'price', 'average_units_sold', 'year_added',
    'min_max_product': ['product_type', 'min_price', 'max_price'],
    'average_price_product': ['product_id', 'price', 'average_units_sold']
}
all_results = []
for output_df_name, cols in requirements.items():
    all_results += safe_check(output_df_name, cols)
check_results_df = pd.DataFrame(all_results)
print(check_results_df)
                  Dataset
                                      Column Exists
                              missing_year 🔽
0
            missing_year
                                product_id
1
              clean_data
2
              clean data
                               product_type
                                                 ~
              clean_data
                                      brand
                                                 \checkmark
3
              clean_data
                                      weight
5
              clean_data
                                      price
                                                 \checkmark
6
              clean_data average_units_sold
                                                 ~
7
              clean_data year_added
                                                  ~
8
              clean data
                           stock_location
                                                  ~
                              product_type
         min_max_product
                                                  ~]
10
         min_max_product
                                  min_price
11
         min_max_product
                                   max_price
                                                  \checkmark
12 average_price_product
                                  product_id
                                                  ~
13 average_price_product
                                     price
                                                  \checkmark
14 average_price_product average_units_sold
                                                  ~
```