

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

Last year (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.

Unknown database ▾ | DataFrame ▾ available as `missing_year`  

```
-- Write your query for task 1 in this cell
SELECT COUNT(*) AS missing_year
FROM products
WHERE year_added IS NULL;
```

	missing_year ▾
0	170

Table | Chart

1 row 

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	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 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 database v DataFrame v available as clean_data

-- Write your query for task 2 in this cell

WITH numeric_weight AS (
  SELECT
    product_id,
    product_type,
    brand,
    price,
    average_units_sold,
    year_added,
    stock_location,
    CASE
      WHEN weight ~ '^[0-9]+(\.[0-9]+)?$' THEN CAST(weight AS numeric)
      ELSE CAST(regexp_replace(weight, '^[0-9.]', '', 'g') AS numeric)
    END AS numeric_weight
  FROM public.products
),
median_weight_cte AS (
  SELECT
    PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY numeric_weight) AS median_weight
  FROM
    numeric_weight
  WHERE
    numeric_weight IS NOT NULL
),
```

```

cleaned_data AS (
  SELECT
    product_id,
    COALESCE(product_type, 'Unknown') AS product_type,
    case when brand = '-' then 'Unknown'
    when brand is null then 'Unknown'
    else brand end AS brand,
    COALESCE(
      ROUND(numeric_weight, 2),
      (SELECT median_weight FROM median_weight_cte)
    ) AS weight,
    COALESCE(price, (SELECT ROUND(AVG(price)::numeric, 2) FROM products WHERE price IS NOT NULL)) AS
price,
    COALESCE(average_units_sold, 0) AS average_units_sold,
    COALESCE(year_added, 2022) AS year_added,
    case when stock_location in ('a', 'b', 'c', 'd') then Upper(stock_location)
    when stock_location is null then 'Unknown'
    else stock_location end AS stock_location
  FROM
    numeric_weight
)
SELECT * FROM cleaned_data;

```



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.

Unknown database ▾ DataFrame ▾ available as min_max_product

-- Write your query for task 3 in this cell

```
SELECT
  product_type,
  MIN(price) AS min_price,
  MAX(price) AS max_price
FROM
  products
GROUP BY
  product_type;
```

	product_type ▾	min_price ▾	max_price ▾
0	Snacks	5.2	10.72
1	Produce	3.46	8.78
2	Dairy	8.33	13.97
3	Bakery	6.26	11.88
4	Meat	11.48	16.98

Table Chart

5 rows ▾

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.

Unknown database ▾

DataFrame ▾

available as `average_price_product`



-- Write your query for task 4 in this cell

```
SELECT
  product_id,
  price,
  average_units_sold
FROM
  products
WHERE
  product_type IN ('Meat', 'Dairy')
  AND average_units_sold > 10;
```

	product_id ▾	price ▾	average_units_sold ▾
0	6	16.2	24
1	8	15.77	28
2	9	11.57	30
3	10	13.94	27
4	11	9.26	26
5	14	11.92	30
6	16	10.79	23
7	19	13.62	26
8	20	13.03	22
9	23	13.07	22

Table Chart

<< < 1 of 70 > >>

Rows per page 10 ▾ 698 rows ↓