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”.

Unknown integration DataFrame as `data_dictionary`

```
SELECT
    column_name,
    data_type,
    is_nullable,
    character_maximum_length
FROM information_schema.columns
WHERE table_name = 'products';
```

index	column_name	data_type	is_nullable	character_maximum_length
0	product_id	integer	YES	
1	product_type	text	YES	
2	brand	text	YES	
3	weight	text	YES	
4	price	real	YES	
5	average_units_sold	integer	YES	
6	year_added	integer	YES	
7	stock_location	text	YES	

Rows: 8 Expand

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 integration DataFrame as `missing_year`

```
SELECT COUNT(*) AS missing_year
FROM products
WHERE year_added IS NULL
```

index	missing_year
0	170

Rows: 1 Expand

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 integration DataFrame as `ra`

```
SELECT
    product_id,
    product_type,
    brand,
    weight,
    price,
    average_units_sold,
    year_added,
    stock_location
FROM products
```

	p...	prod...	brand	weig...		average_units_...	y...	stock_lo...
0	1	Bakery	TopBrand	602.61 grams	11	15		C
1	2	Produce	SilverLake	478.26	8.08	22	2022	C
2	3	Produce	TastyTreat	532.38 grams	6.16	21	2018	B
3	4	Bakery	StandardYums	453.43 grams	7.26	21	2021	d
4	5	Produce	GoldTree	588.63	7.88	21	2020	a
5	6	Meat	TopBrand	612.06	16.2	24	2017	a
6	7	Produce	GoldTree	320.49	8.01	21	2019	B
7	8	Meat	SilverLake	535.19 grams	15.77	28	2021	a
8	9	Meat	StandardYums	375.07 grams	11.57	30	2020	a
9	10	Meat	TastyTreat	506.34	13.94	27	2018	C
10	11	Dairy	StandardYums	345.07	9.26	26	2020	b
11	12	Bakery	StandardYums	345.58	6.87	21	2022	D
12	13	Snacks	SmoothTaste	512.54 grams	8.65	19	2016	A
13	14	Meat	StandardYums	395.76 grams	11.92	30	2019	A
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

Expand

Unknown integration DataFrame as c

```
SELECT
    product_id,
    CASE WHEN product_type IS NULL THEN 'Unknown' ELSE product_type END AS product_type,
    CASE WHEN brand IS NULL THEN 'Unknown' WHEN brand = '-' THEN 'Unknown' ELSE brand END AS brand,
    CASE WHEN weight IS NULL THEN (
        SELECT ROUND(PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY CAST(REPLACE(weight, ' grams', '') AS NUMERIC)), 2)
        FROM products
    )
    ELSE ROUND(CAST(REPLACE(weight, ' grams', '') AS NUMERIC), 2)
    END AS weight,
    CASE WHEN price IS NULL THEN (
        SELECT ROUND(PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY price)::numeric, 2)
        FROM products
    ) ELSE ROUND(price::numeric, 2) END AS price,
    CASE WHEN average_units_sold IS NULL THEN 0 ELSE average_units_sold END AS average_units_sold,
    CASE WHEN year_added IS NULL THEN 2022 ELSE year_added END AS year_added,
    CASE WHEN stock_location IS NULL THEN 'Unknown' ELSE UPPER(stock_location) END AS stock_location
FROM products
```

	p...	prod...	brand			average_units_...	y...	stock_lo...
0	1	Bakery	TopBrand	602.61	11	15	2022	C
1	2	Produce	SilverLake	478.26	8.08	22	2022	C
2	3	Produce	TastyTreat	532.38	6.16	21	2018	B
3	4	Bakery	StandardYums	453.43	7.26	21	2021	D
4	5	Produce	GoldTree	588.63	7.88	21	2020	A
5	6	Meat	TopBrand	612.06	16.2	24	2017	A
6	7	Produce	GoldTree	320.49	8.01	21	2019	B
7	8	Meat	SilverLake	535.19	15.77	28	2021	A
8	9	Meat	StandardYums	375.07	11.57	30	2020	A
9	10	Meat	TastyTreat	506.34	13.94	27	2018	C
10	11	Dairy	StandardYums	345.07	9.26	26	2020	B
11	12	Bakery	StandardYums	345.58	6.87	21	2022	D
12	13	Snacks	SmoothTaste	512.54	8.65	19	2016	A
13	14	Meat	StandardYums	395.76	11.92	30	2019	A
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 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.

Unknown integration DataFrame as `m`

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

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

Rows: 5 Expand

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 integration DataFrame as

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

	p...		average_units_...
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
10	24	10.98	23
11	25	12.81	24
12	28	13.01	20
13	31	13.11	20
14	41	8.63	27
15	42	12.56	24

Rows: 698

Expand