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 integration DataFrame as `missing_year`

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

index	missing_year
0	

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 clean_data

```sql
-- Write your query for task 2 in this cell
WITH cleaned_products AS (
    SELECT *,
        CAST(NULLIF(REGEXP_REPLACE(weight::text, '[^0-9\.]', '', 'g'), '') AS NUMERIC) AS cleaned_weight,
        CAST(NULLIF(REGEXP_REPLACE(price::text, '[^0-9\.]', '', 'g'), '') AS NUMERIC) AS cleaned_price
    FROM products
),
medians AS (
    SELECT
        PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY cleaned_weight) AS median_weight,
        PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY cleaned_price) AS median_price
    FROM cleaned_products
)

SELECT
    product_id,
    COALESCE(product_type, 'Unknown') AS product_type,
    CASE
        WHEN brand IS NULL OR brand = '-' THEN 'Unknown'
        ELSE brand
    END AS brand,
    COALESCE(cleaned_weight, m.median_weight) AS weight,
    COALESCE(cleaned_price, m.median_price) AS price,
    COALESCE(average_units_sold, 0) AS average_units_sold,
    COALESCE(year_added, 2022) AS year_added,
    CASE
        WHEN stock_location IS NULL THEN 'Unknown'
        ELSE UPPER(stock_location)
    END AS stock_location
FROM cleaned_products cp, medians m;
```

	p...	prod...	brand			average_units_...	y...	stock_lo...
4	5	Produce	GoldTree	588.63	7.88	21	2020	A
6	7	Produce	GoldTree	320.49	8.01	21	2019	B
19	20	Dairy	GoldTree	360.76	13.03	22	2019	B
21	22	Produce	GoldTree	440.03	8.18	21	2022	D
22	23	Dairy	GoldTree	533.58	13.07	22	2021	C
37	38	Produce	GoldTree	465.35	7.87	21	2018	A
42	43	Produce	GoldTree	591.3	8.15	21	2018	A
44	45	Bakery	GoldTree	635.66	11.1	16	2017	A
45	46	Produce	GoldTree	545.98	7.85	21	2015	C
48	49	Produce	GoldTree	391.65	7.95	21	2022	A
49	50	Dairy	GoldTree	548.67	12.75	22	2022	C
54	55	Produce	GoldTree	364.23	7.99	21	2017	A
61	62	Meat	GoldTree	503	16.09	26	2019	D
66	67	Produce	GoldTree	502.6	8.16	21	2016	D
73	74	Produce	GoldTree	591.88	8.1	21	2022	A
76	77	Meat	GoldTree	593.72	16.34	25	2021	A

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

```sql
-- Write your query for task 3 in this cell
WITH cleaned_products AS (
    SELECT
        COALESCE(product_type, 'Unknown') AS product_type,
        CAST(NULLIF(REGEXP_REPLACE(price::text, '[^0-9\.]', '', 'g'), '') AS NUMERIC) AS cleaned_price
    FROM products
)

SELECT
    product_type,
    MIN(cleaned_price) AS min_price,
    MAX(cleaned_price) AS max_price
FROM cleaned_products
GROUP BY product_type
ORDER BY product_type;
```

	prod...	m	m
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 a

```
-- Write your query for task 4 in this cell
WITH cleaned_products AS (
    SELECT
        product_id,
        product_type,
        CAST(NULLIF(REGEXP_REPLACE(price::text, '[^0-9\.]', '', 'g'), '') AS NUMERIC) AS cleaned_price,
        COALESCE(average_units_sold, 0) AS average_units_sold
    FROM products
)

SELECT
    product_id,
    cleaned_price AS price,
    average_units_sold
FROM cleaned_products
WHERE product_type IN ('Meat', 'Dairy')
  AND average_units_sold > 10
ORDER BY product_id;
```

	p...		average_units_...
12	28	13.01	20
13	31	13.11	20
16	47	12.86	20
50	141	12.82	20
61	172	12.87	20
72	197	12.75	20
87	238	12.64	20
135	341	13.12	20
164	422	12.99	20
167	433	12.79	20
168	434	13.36	20
172	448	12.79	20
174	462	12.7	20
207	551	12.61	20
226	587	13.15	20
247	637	13.28	20

Rows: 698 Expand