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. </br>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). </br>Missing values should be replaced with “Unknown”.
brand	Nominal. The brand of the product. One of 7 possible values. </br>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. </br>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. </br>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. </br>Missing values should be replaced with 0.
year_added	Nominal. The year the product was first added to FoodYum stock.</br>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 </br>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.

In [31]:

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

Out[31]:

	missing_year
0	170

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. </br>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). </br>Missing values should be replaced with "Unknown".
brand	Nominal. The brand of the product. One of 7 possible values. </br>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. </br>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. </br>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. </br>Missing values should be replaced with 0.
year_added	Nominal. The year the product was first added to FoodYum stock.</br>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 </br>Missing values should be replaced with "Unknown".

In [32]:

```
-- Write your query for task 2 in this cell
CREATE TEMP TABLE clean_data AS
SELECT product_id,
 CASE WHEN product_type = '-' OR product_type IS NULL THEN 'Unknown' ELSE product_type E
ND,
 CASE WHEN brand = '-' OR brand IS NULL THEN 'Unknown' ELSE brand END,
   COALESCE(
        CASE
            WHEN regexp_replace(weight, '[^0-9.]', '', 'g') = '' THEN (SELECT PERCENTILE
_CONT(0.5) WITHIN GROUP (ORDER BY CAST(regexp_replace(weight, '[^0-9.]', '', 'g') AS NUM
ERIC)) FROM products) ELSE CAST(regexp_replace(weight, '[^0-9.]', '', 'g') AS NUMERIC)EN
D,
        (SELECT PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY CAST(regexp_replace(weight,
'[^0-9.]', '', 'g') AS NUMERIC)) FROM products)) AS weight,
    COALESCE(price, (SELECT PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY price) FROM prod
ucts)) AS price,
 COALESCE(average_units_sold, 0) AS average_units_sold,
 COALESCE (year_added, 2022) AS year_added,
 CASE WHEN
stock_location = 'a' THEN 'A'
  WHEN stock_location = 'b' THEN 'B'
  WHEN stock_location = 'c' THEN 'C'
  WHEN stock_location = 'd' THEN 'D'
  WHEN stock_location IS NULL THEN 'Unknown'
  ELSE stock_location END AS stock_location
FROM products;

SELECT  product_id,
    product_type,
    brand,
    CASE
        WHEN weight IS NOT NULL THEN ROUND(weight::numeric, 2) ELSE NULL END AS weight,
     ROUND(CAST(price AS numeric), 2) AS price,
  average_units_sold,
  year_added,
    stock_location
FROM clean_data;
```

Out[32]:

```
        product_id  product_type            brand  weight  price  average_units_sold  year_added  stock_location
```

	product_id	product_type	brand	weight	price	average_units_sold	year_added	stock_location
0	1	Bakery	TopBrand	602.61	11.00	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
...	...	...	...	...	...	...	...	...
1695	1696	Meat	TastyTreat	503.99	14.08	25	2017	A
1696	1697	Meat	GoldTree	526.89	16.13	25	2016	D
1697	1698	Bakery	YumMie	583.85	7.05	16	2021	A
1698	1699	Produce	TopBrand	441.64	8.10	19	2019	A
1699	1700	Meat	TopBrand	518.60	15.89	24	2021	A

1700 rows × 8 columns

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.

In [33]:

```
-- 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;
```

Out[33]:

	product_type	min_price	max_price
0	Snacks	5.20	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

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.

In [34]:

```
-- Write your query for task 4 in this cell
SELECT product_id,
price,
average_units_sold
FROM (SELECT product_id, product_type, price, average_units_sold
  FROM products
  WHERE product_type = 'Meat' OR product_type = 'Dairy') AS filter
WHERE average_units_sold > 10;
```

Out[34]:

	product_id	price	average_units_sold
0	6	16.20	24
1	8	15.77	28
2	9	11.57	30
3	10	13.94	27
4	11	9.26	26
...	...	...	...
693	1694	16.00	25
694	1695	12.88	20
695	1696	14.08	25
696	1697	16.13	25
697	1700	15.89	24

698 rows × 3 columns