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.

Certification Grocery Store 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;

-- select * from missing_year;


-- select * from products;
```

index	missing_year
0	

Rows: 1

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”.

Certification Grocery Store DataFrame as c

```
-- Write your query for task 2 in this cell
-- update products
-- REGEXP_REPLACE(column_name, '[^0-9.]', '', 'g')

-- with median_weight as(
--  select percentile_cont(0.5) within group (order by weight::NUMERIC) as median_weight from products),

-- median_price as(
--  select percentile_cont(0.5) within group (order by price::NUMERIC) as median_price from products),


-- clean_data as(
--  select cast(product_id as varchar(8)) as product_id,
--  case when product_type is null then 'Unknown' else cast(product_type as varchar(10)) end as product_type,
--  case when brand is null then 'Unknown' else cast(brand as varchar(15)) end as brand,
--  coalesce(cast(weight as numeric(3, 2)), median_weight) as weight,
--  coalesce(cast(price as numeric(2, 2)), median_price) as price
--  from products
--     CROSS JOIN median_weight
--     CROSS JOIN median_price)

-- select * from clean_data;


WITH median_weight AS (
    SELECT percentile_cont(0.5) WITHIN GROUP (ORDER BY (REGEXP_REPLACE(weight::TEXT, '[^0-9.]', '', 'g')::NUMERIC)) AS median_weight
    FROM products
),
median_price as(
    select percentile_cont(0.5) within group (order by price::NUMERIC) as median_price from products
),
clean_data AS (
    SELECT
        CAST(product_id AS VARCHAR(8)) AS product_id,
        COALESCE(CAST(product_type AS VARCHAR(10)), 'Unknown') AS product_type,
        COALESCE(
    CASE
        WHEN brand IN ('-', 'missing') OR brand IS NULL THEN 'Unknown'
        ELSE brand
    END,
    'Unknown') AS brand,
        COALESCE(
            CAST(REGEXP_REPLACE(weight::TEXT, '[^0-9.]', '', 'g') AS NUMERIC),
            median_weight
        ) AS weight,
        coalesce(cast(price as numeric), median_price) as price,
        coalesce(cast(average_units_sold as integer), 0) as average_units_sold,
        coalesce(cast(year_added as integer), 2022) as year_added,
        coalesce(upper(cast(stock_location as char(1))), 'Unknown') as stock_location
    FROM products
    CROSS JOIN median_weight
```

```
    CROSS JOIN median_price
)
SELECT * FROM clean_data;
```

	p...	prod...	brand			average_units_...	y...	stock_lo...
32	33	Snacks	YumMie	445.79	6.49	18	2017	D
36	37	Snacks	YumMie	299.46	5.81	18	2018	B
53	54	Bakery	YumMie	559.66	7.01	16	2015	C
80	81	Bakery	YumMie	401.34	6.85	16	2019	A
98	99	Snacks	YumMie	423.91	6.4	18	2017	B
104	105	Bakery	YumMie	487.64	6.87	16	2020	A
155	156	Bakery	YumMie	454.11	7.19	16	2020	D
168	169	Snacks	YumMie	419	6.11	17	2019	A
199	200	Bakery	YumMie	614.85	7.36	16	2016	C
213	214	Snacks	YumMie	408.17	6.23	18	2019	B
220	221	Bakery	YumMie	427.91	7.1	16	2019	A
228	229	Meat	YumMie	420.59	12.04	25	2021	B
268	269	Produce	YumMie	538.12	4.08	21	2021	C
292	293	Dairy	YumMie	697.9	9.07	22	2021	B
298	299	Snacks	YumMie	477.15	5.86	18	2022	C
307	308	Dairy	YumMie	577.78	8.95	22	2022	B

Rows: 1,700

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.

Certification Grocery Store DataFrame as `m`

```
-- Write your query for task 3 in this cell
-- select * from products;
with min_max_product as(
select product_type, min(price) as min_price, max(price) as max_price from products
group by product_type)

select * from min_max_product;
```

	prod...	m	m
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

Rows: 5

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.

Certification Grocery Store DataFrame as

```
-- Write your query for task 4 in this cell
with average_price_product as(
select product_id, price, average_units_sold from products
where product_type in ('Meat', 'Dairy') and average_units_sold>10)

select * from average_price_product;
```

	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