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 `Overview`

```
--- Overview of products table

SELECT *
FROM products
LIMIT 10;
```

	p…	prod…	brand	weig…	…	average_units_…	y…	stock_loc…
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

Rows: 10 Expand

Unknown integration DataFrame as `product_id`

```
--- Exploring product_id column for NULLs, empty strings, missing, or inconsistent data

SELECT product_id
FROM products
WHERE product_id IS NULL;
```

Your query ran successfully but returned no results.

Unknown integration DataFrame as `product_type`

```
--- Exploring product_type column for NULLs, empty strings, missing, or inconsistent data

SELECT DISTINCT product_type
FROM products;
```

	prod…
0	Snacks
1	Produce
2	Dairy
3	Bakery
4	Meat

Rows: 5 Expand

Unknown integration DataFrame as `bra`

```
--- Exploring brand column for NULLs, empty strings, missing, or inconsistent data

SELECT DISTINCT brand
FROM products;
```

	brand
0	StandardYums
1	SmoothTaste
2	TastyTreat
3	-
4	YumMie
5	SilverLake
6	GoldTree
7	TopBrand

Rows: 8 Expand

Unknown integration DataFrame as

```
--- Exploring the weight column for NULLs, empty strings, missing, or inconsistent data

SELECT
    weight
FROM products
WHERE
    weight IS NULL;
```

Your query ran successfully but returned no results.

Unknown integration DataFrame as

```
--- Exploring price column for NULLs, empty strings, missing, or inconsistent data

SELECT
    price
FROM products
WHERE
    price IS NULL;
```

Your query ran successfully but returned no results.

Unknown integration DataFrame as

```
--- Looking for Max, Min, and Avg prices in our dataset

SELECT
    AVG(price),
    MAX(price),
    MIN(price)
FROM products;
```

	avg		
0	10.1317235325	16.98	3.46

Rows: 1 Expand

Unknown integration DataFrame as

```
--- Exploring the average_units_sold column for NULLs, empty strings, missing, or inconsistent data

 SELECT
    average_units_sold
FROM products
WHERE
    average_units_sold IS NULL;
```

Your query ran successfully but returned no results.

Unknown integration DataFrame as

```
--- Looking for Max, Min, and Avg average_units_sold in our dataset

SELECT
    MAX(average_units_sold),
    MIN(average_units_sold)
FROM products;
```

0	31	14

Rows: 1 Expand

Unknown integration DataFrame as

```
--- Exploring the year_added column for NULLs, empty strings, missing, or inconsistent data

SELECT DISTINCT year_added
FROM products;
```

	y…
0	2021
1	2020
2	2015
3	
4	2022
5	2017
6	2016
7	2019
8	2018

Rows: 9 Expand

Unknown integration DataFrame as

```
--- Exploring the stock location column for NULLs, empty strings, missing, or inconsistent data

SELECT DISTINCT stock_location
FROM products;
```

	stock_lo…
0	B
1	a
2	D
3	A
4	d
5	C
6	c
7	b

Rows: 8 Expand

Unknown integration DataFrame as

```
--- Understanding the table structure and data types of all the columns

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

	column_name	d
0	product_id	integer
1	product_type	text
2	brand	text
3	weight	text
4	price	real
5	average_units_sold	integer
6	year_added	integer
7	stock_location	text

Rows: 8 Expand

We have checked each and every columns indivisually and here is what we found:

1. product-id: PERFECT! No NULLS no data types errors!!
2. product-type: PERFECT! No NULLS no data type errors!!
3. brand: NULLS as '-', Capitalization issues, space issues
4. weight: NO NULLS, datatype issues, conatins 'grams' in few rows
5. price: PERFECT!! NO NULLS No datatype errors!!
6. average_units_sold: PERFECT!! NO NULLS No datatype errors!!
7. year_added: NULL as Missing value
8. stock-location: Sames location diff capitalization

Task 1

In 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

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

	missi...
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

```sql
-- Write your query for task 2 in this cell


--- Creating a CTE to replace NULLS in weight with Median value
WITH weight_median AS(
    SELECT
        PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY REPLACE(weight, ' grams', '')::NUMERIC) AS median_weight
    FROM products
    WHERE weight IS NOT NULL
),

--- Creating a CTE to replace NULLS in price with Median value
price_median AS (
    SELECT
        PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY price :: NUMERIC) AS median_price
    FROM products
    WHERE price IS NOT NULL
)

-------------------------------------------------------------------------------------

SELECT

    --- NO CHANGES
    product_id :: INTEGER,

    --- NO CHANGES
    COALESCE(TRIM(product_type) :: TEXT, 'Unknown') AS product_type,

    --- Replace NULL and '-' in brands with 'Unknown'
    COALESCE(NULLIF(TRIM(brand), '-'), 'Unknown') AS brand,

    --- Replace NULLS with 'Median' from CTE, Cast type Numeric, round to 2 deci places
    ROUND(
        COALESCE(
            REPLACE(weight, ' grams', '') :: NUMERIC,
            (SELECT median_weight FROM weight_median)
        ) :: NUMERIC, 2) AS weight,

    --- Replace NULLS with 'Median' from CTE, Cast type Numeric, round to 2 deci places
    ROUND(COALESCE(price, (SELECT median_price FROM price_median)) :: NUMERIC, 2) AS price,

    --- Replace NULLS with '0' and cast type Integer
    COALESCE(average_units_sold :: INTEGER, 0) As average_units_sold,

    --- Replace NULLS with recent year '2022' and cast type to Integer
    COALESCE(
        CASE
            WHEN year_added :: TEXT ~ '^\d+$' THEN year_added::INTEGER
            ELSE NULL
        END,
    2022) AS year_added,

    --- Normalized capitalization and replace invalid or NULLS with 'Unknown'
    CASE
        WHEN UPPER(stock_location) IN ('A', 'B', 'C', 'D') THEN UPPER(stock_location)
        ELSE 'Unknown'
    END AS stock_location

FROM public.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

```
-- Write your query for task 3 in this cell
SELECT
    product_type,
    MIN(price) AS min_price,
    MAX(price) AS max_price
FROM public.products
GROUP BY product_type;
```

	prod...		
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 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

```sql
-- Write your query for task 4 in this cell
SELECT
    product_id,
    price,
    average_units_sold
FROM
    public.products
WHERE
   product_type IN ('Meat', 'Dairy') 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

FORMATTING AND NAMING CHECK

Use the code block below to check that your outputs are correctly named and formatted before you submit your project.

This code checks whether you have met our automarking requirements: that the specified DataFrames exist and contain the required columns. It then prints a table showing ✅ for each column that exists and ❌ for any that are missing, or if the DataFrame itself isn't available.

If a DataFrame or a column in a DataFrame doesn't exist, carefully check your code again.

IMPORTANT: even if your code passes the check below, this does not mean that your entire submission is correct. This is a check for naming and formatting only.

```
import pandas as pd

def check_columns(output_df, output_df_name, required_columns):
    results = []
    for col in required_columns:
        exists = col in output_df.columns
        results.append({'Dataset': output_df_name, 'Column': col, 'Exists': '✅' if exists else '❌'})
    return results

def safe_check(output_df_name, required_columns):
    results = []
    if output_df_name in globals():
        obj = globals()[output_df_name]
        if isinstance(obj, pd.DataFrame):
            results.extend(check_columns(obj, output_df_name, required_columns))
        elif isinstance(obj, str) and ("SELECT" in obj.upper() or "FROM" in obj.upper()):
            results.append({'Dataset': output_df_name, 'Column': '—', 'Exists': 'ℹ️ SQL query string'})
        else:
            results.append({'Dataset': output_df_name, 'Column': '—', 'Exists': '❌ Not a DataFrame or query'})
    else:
        results.append({'Dataset': output_df_name, 'Column': '—', 'Exists': '❌ Variable not defined'})
    return results

requirements = {
    'missing_year': ['missing_year'],
    'clean_data': ['product_id', 'product_type', 'brand', 'weight', 'price', 'average_units_sold', 'year_added',
'stock_location'],
    'min_max_product': ['product_type', 'min_price', 'max_price'],
    'average_price_product': ['product_id', 'price', 'average_units_sold']
}

all_results = []
for output_df_name, cols in requirements.items():
    all_results += safe_check(output_df_name, cols)

check_results_df = pd.DataFrame(all_results)

print(check_results_df)
```

	Dataset	Column	Exists
0	missing_year	missing_year	✅
1	clean_data	product_id	✅
2	clean_data	product_type	✅
3	clean_data	brand	✅
4	clean_data	weight	✅
5	clean_data	price	✅
6	clean_data	average_units_sold	✅
7	clean_data	year_added	✅
8	clean_data	stock_location	✅
9	min_max_product	product_type	✅
10	min_max_product	min_price	✅
11	min_max_product	max_price	✅
12	average_price_product	product_id	✅
13	average_price_product	price	✅
14	average_price_product	average_units_sold	✅