Grocery Price Analysis

Project Goal

The goal of this project is to implement various SQL queries on a Fruits and Vegetables Prices in USA dataset to return interesting insights about the dataset. This dataset includes information such as item names, forms, retail prices etc. This project aims to uncover interesting facts and patterns within the data, providing valuable information related to the economic aspects of various food items. This analysis could be used to understand pricing trends, yield variations and other factors in this domain.

This project provides a platform to apply and test key concepts learned in class like:

- Demonstrating the ability to design a conceptual model that represents the relationships and entities within the real-world food dataset.
- Translation the conceptual model into SQL schema by creating appropriate tables. This involves mapping the conceptual entities and relationships to their corresponding database structures.
- Implementing the SQL schema by creating tables that accurately represent the dataset. This involves considering data types, constraints and relationships.
- Demonstrating proficiency in writing SQL queries tailored for specific data retrieval purposes. This includes extracting meaningful insights from the dataset to address relevant economic questions.

Attached Files

- Raw dataset used: .csv
- Transformed datasets: prices.csv and groceries.csv
- Relations in schema: item, retail_price

Dataset

We downloaded the dataset from Kaggle

Dataset: Fruits And Vegetables Prices In USA

Link: https://www.kaggle.com/datasets/anshikakashyap12/fruits-and-vegetables-prices-in-usa/

This dataset contains information about the 'Fruits and Vegetables Prices In USA'. This dataset contains 8 columns and 156 rows.

It contains columns like Item(Name), form(Canned, fresh, dried and frozen), Retail Price etc..

Here is a snapshot of the raw data from dataset.csv “Prices.csv”, the dataset has 8 columns.

	A	B	C	D	E	F	G	H	I
1	Item	Form	RetailPrice	RetailPriceUnit	Yield	CupEquivalentSize	CupEquivalentUnit	CupEquivalentPrice	
2	Acorn squash	Fresh	1.1804	per pound	0.4586	0.4519	pounds	1.1633	
3	Apples	Fresh	1.5193	per pound	0.9	0.2425	pounds	0.4094	
4	Apples, applesauce	Canned	1.066	per pound	1	0.5401	pounds	0.5758	
5	Apples, frozen concentrat	Juice	0.5853	per pint	1	8	fluid ounces	0.2926	
6	Apples, ready-to-drink	Juice	0.7804	per pint	1	8	fluid ounces	0.3902	
7	Apricots	Fresh	2.9665	per pound	0.93	0.3638	pounds	1.1603	
8	Apricots	Dried	6.6188	per pound	1	0.1433	pounds	0.9485	
9	Apricots, packed in juice	Canned	1.6905	per pound	1	0.5401	pounds	0.9131	
10	Apricots, packed in syrup	Canned	2.06	per pound	0.65	0.4409	pounds	1.3974	
11	Artichoke	Fresh	2.1913	per pound	0.375	0.3858	pounds	2.2545	

Data Transformation

In order to perform SQL queries on our dataset, we have divided the original data table into two distinct tables.

We strategically divided in the data into two distinct tables: “Groceries” and “prices”.

I.e. “Groceries.csv” and “prices.csv”

- **Here unit id mentioned as “uid”**

Snapshot for Groceries.csv

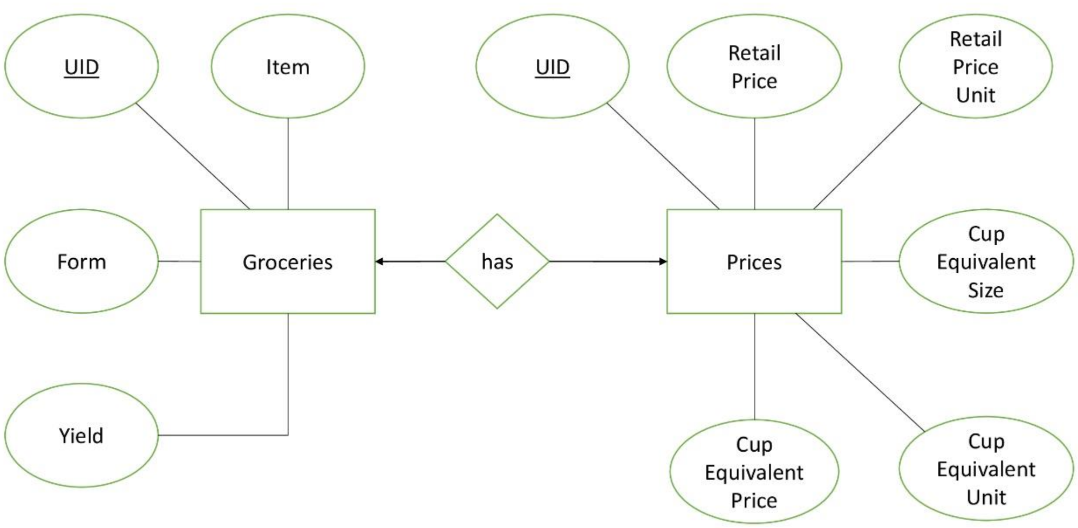
	uid integer	item character (64)	form character (64)	yield double precision
1	1	Acorn squash	Fresh ...	0.4586
2	2	Apples	Fresh ...	0.9
3	3	Apples, applesauce	Canned ...	1
4	4	Apples, frozen concentrate	Juice ...	1
5	5	Apples, ready-to-drink	Juice ...	1
6	6	Apricots	Fresh ...	0.93
7	7	Apricots	Dried ...	1
8	8	Apricots, packed in juice	Canned ...	1
9	9	Apricots, packed in syrup or water ...	Canned ...	0.65
10	10	Artichoke	Fresh ...	0.375

Snapshot for prices.csv:

	uid integer	retailprice double precision	retailpriceunit character (64)	cupequivalentsize double precision	cupequivalentunit character (64)	cupequivalentprice double precision
1	1	1.1804	per pound	0.4519	pounds	1.1633
2	2	1.5193	per pound	0.2425	pounds	0.4094
3	3	1.066	per pound	0.5401	pounds	0.5758
4	4	0.5853	per pint	8	fluid ounces	0.2926
5	5	0.7804	per pint	8	fluid ounces	0.3902
6	6	2.9665	per pound	0.3638	pounds	1.1603
7	7	6.6188	per pound	0.1433	pounds	0.9485
8	8	1.6905	per pound	0.5401	pounds	0.9131
9	9	2.06	per pound	0.4409	pounds	1.3974
10	10	2.1913	per pound	0.3858	pounds	2.2545

Conceptual Design

The ER (Entity-Relationship) diagram provided comprehensively details the specifications of our data, offering a holistic representation of the interrelationships between entities and their attributes within our database schema.



Database Schema – DDL Statements

We converted the above conceptual design into the following SQL Schema:

```
Create table Groceries(
      UId int NOT NULL,
      Item char(64) NOT NULL,
      Form char(64) NOT NULL,
      Yield float(32) NOT NULL
);
Create table prices(
      UId int NOT NULL,
   RetailPrice float NOT NULL,
      RetailPriceUnit Char(64) NOT NULL,
  CupEquivalentSize float(32) NOT NULL,
      CupEquivalentUnit Char(64) NOT NULL,
      CupEquivalentPrice float NOT NULL
);
```

DML Statements

We populated our schema with the following DML statements:

```
INSERT INTO  Groceries (Item,Form,Yield)

INSERT INTO
prices(RetailPrice,RetailPriceUnit,CupEquivalentSize,CupEquivalentUnit
,CupEquivalentPrice)


 COPY Groceries(Item,Form,Yield)

FROM 'C:\Users\heman\Downloads\prices.csv' DELIMITER ','

CSV HEADER;

COPY prices(RetailPrice,RetailPriceUnit
,CupEquivalentSize,CupEquivalentUnit,CupEquivalentPrice)

FROM 'C:\Users\heman\Downloads\prices.csv' DELIMITER ','

CSV HEADER;
```

Methodology

After organizing the database schema into distinct relations tailored to our project's needs, specifically focusing on entities such as 'groceries' and 'prices' we proceeded to implement SQL queries designed for targeted and scenario-based data retrieval. These queries are written to address specific requirements and scenarios within the scope of our grocery management system, allowing us to efficiently retrieve relevant information for analysis and decision-making.

Let's see what are the Fresh form of Groceries and vegetables available

```
SELECT item, form FROM Groceries

WHERE form = 'Fresh';
```

Data Output Explain Notifications

	item character (64)	form character (64)
1	Acorn squash	Fresh
2	Apples	Fresh
3	Apricots	Fresh
4	Artichoke	Fresh
5	Asparagus	Fresh
6	Avocados	Fresh
7	Bananas	Fresh
8	Blackberries	Fresh
9	Blueberries	Fresh
10	Broccoli florets	Fresh
11	Broccoli heads	Fresh
12	Brussels sprouts	Fresh
13	Butternut squash	Fresh
14	Cabbage, green	Fresh
15	Cabbage, red	Fresh
16	Cantaloupe	Fresh
17	Carrots, baby	Fresh
18	Carrots, cooked whole	Fresh
19	Carrots, raw whole	Fresh
20	Cauliflower florets	Fresh

- This output will consist of the selected columns from the matching rows in the "Groceries" table that are fresh.

What is the correlation between the form of fruits and vegetables and its retail price

```
Select form, AVG(RetailPrice) as avg_retail_price

from Groceries

inner join prices on Groceries.UId = prices.Uid

Group By form;
```

	form character (64)	avg_retail_price double precision
1	Fresh	2.03828636363636
2	Canned	1.66583611111111
3	Juice	1.10855454545455
4	Dried	4.06167647058824
5	Frozen	2.483644

➢ From output, we can understand that each row represents a unique form along with its average retail price.

What is the relation between the yield of the fruits or vegetables and its retail price

```
SELECT item,yield,RetailPrice FROM Groceries

INNER JOIN prices ON Groceries.UId = prices.Uid;
```

	item character (64)	yield double precision	retailprice double precision
1	Acorn squash	0.4586	1.1804
2	Apples	0.9	1.5193
3	Apples, applesauce	1	1.066
4	Apples, frozen concentrate	1	0.5853
5	Apples, ready-to-drink	1	0.7804
6	Apricots	0.93	2.9665
7	Apricots	1	6.6188
8	Apricots, packed in juice	1	1.6905
9	Apricots, packed in syrup or...	0.65	2.06
10	Artichoke	0.375	2.1913
11	Artichoke	0.65	3.4119
12	Asparagus	0.4938	2.7576
13	Asparagus	0.65	3.1269
14	Asparagus	1.0335	6.7045
15	Avocados	0.7408	2.2368
16	Bananas	0.64	0.5249
17	Beets	0.65	1.2684
18	Berries, mixed	1	3.5585
19	Black beans	0.65	1.0281
20	Black beans	2.4692	1.3753

➢ From the result we can understand the set of rows where each row contains information about a grocery item's name, yield, and retail price.

How does the cup equivalent price of Groceries or vegetables compare across different forms

```
SELECT form, MAX(CupEquivalentPrice) AS avg_cup_equi
FROM Groceries
INNER JOIN prices ON Groceries.UId = prices.Uid
GROUP BY form
ORDER BY avg_cup_equi;
```

Data Output Explain Notifications

	form character (64)	avg_cup_equi double precision
1	Dried	1.3219
2	Juice	1.561
3	Fresh	2.2545
4	Frozen	2.5742
5	Canned	3.07

- From the output, we get to know the maximum cup equivalent price for each form of groceries.

Which grocery item offers a higher cost-effectiveness per cup, taking into account both retail price and yield?

```
SELECT
    g.UId,
    g.Item,
    (p.RetailPrice / (g.Yield * p.CupEquivalentSize)) AS
EffectivePricePerCup
FROM
    Groceries g
JOIN
    Prices p ON g.UId = p.UId
ORDER BY
    EffectivePricePerCup;
```

	uid integer	item character (64)	effectivepricepercup double precision
1	4	Apples, frozen concentrate	0.0731625
2	121	Pineapple, frozen concentrate ...	0.0871625
3	67	Grapes, frozen concentrate	0.0889875
4	108	Oranges, frozen concentrate	0.096125
5	5	Apples, ready-to-drink	0.09755
6	68	Grapes, ready-to-drink	0.1151875
7	109	Oranges, ready-to-drink	0.123025
8	124	Pineapple, ready-to-drink	0.1286
9	64	Grapefruit, ready-to-drink	0.1301875
10	128	Plum (prune), ready-to-drink	0.194025

- From the output we get to know that the effective price per cup indicates the cost-effectiveness of each grocery item in terms of its retail price, yield, and cup equivalent size. Items with lower effective prices per cup are more cost-effective in this context.

What variations occur in the price per cup for different quantities of each item?

```
WITH CalculatedPrices AS (
    SELECT
        Groceries.UId,
        Groceries.Item,
        prices.CupEquivalentUnit,
        prices.CupEquivalentSize,
        prices.RetailPrice,
        Groceries.Yield,
        prices.RetailPrice / (Groceries.Yield *
prices.CupEquivalentSize) AS PricePerCup
    FROM
        Groceries
    JOIN
        Prices  ON Groceries.UId = prices.UId
)
SELECT
    UId,
    Item,
```

```
        CupEquivalentUnit,
        CupEquivalentSize,
        Yield,
        RetailPrice,
        PricePerCup,
        ROW_NUMBER() OVER (PARTITION BY UId ORDER BY PricePerCup) AS
    ScenarioRank
    FROM
        CalculatedPrices;
```

	uid integer	item character (64)	cupequivalentunit character (64)	cupequivalentsize double precision	yield double precision	retailprice double precision	pricepercup double precision	scenariorank bigint
1	1	Acorn squash	pounds	0.4519	0.4586	1.1804	5.69577479087908	1
2	2	Apples	pounds	0.2425	0.9	1.5193	6.96128293241695	1
3	3	Apples, applesauce	pounds	0.5401	1	1.066	1.97370857248658	1
4	4	Apples, frozen concentrate	fluid ounces	8	1	0.5853	0.0731625	1
5	5	Apples, ready-to-drink	fluid ounces	8	1	0.7804	0.09755	1
6	6	Apricots	pounds	0.3638	0.93	2.9665	8.76796301879208	1
7	7	Apricots	pounds	0.1433	1	6.6188	46.1884159106769	1
8	8	Apricots, packed in juice	pounds	0.5401	1	1.6905	3.12997593038326	1
9	9	Apricots, packed in syrup or water	pounds	0.4409	0.65	2.06	7.18809428267355	1
10	10	Artichoke	pounds	0.3858	0.375	2.1913	15.1463625367202	1

- ➢ From the output we can understand variations in the price per cup across different quantities. The scenariorank column can help us in understanding how each item ranks in terms of cost-effectiveness per cup within its own group.

What is the average, minimum and maximum retail prices, along with the priceobservations for each item?

```
SELECT
    Groceries.Item,
    AVG(prices.RetailPrice) AS AveragePrice,
    MIN(prices.RetailPrice) AS MinPrice,
    MAX(prices.RetailPrice) AS MaxPrice,
    COUNT(*) AS PriceObservations
FROM Groceries
JOIN
    Prices  ON Groceries.UId = prices.UId
GROUP BY
    Groceries.Item
ORDER BY
```

```
    AveragePrice DESC;
```

	item character (64)	averageprice double precision	minprice double precision	maxprice double precision	priceobservations bigint
1	Figs	6.8371	6.8371	6.8371	1
2	Mangoes	5.852	1.1513	10.5527	2
3	Olives	5.7719	5.7719	5.7719	1
4	Plum (prunes)	5.7042	5.7042	5.7042	1
5	Dates	5.5713	5.5713	5.5713	1
6	Raspberries	5.4134	4.1877	6.6391	2
7	Blackberries	4.8267	3.6362	6.0172	2
8	Apricots	4.79265	2.9665	6.6188	2
9	Cranberries	4.6513	4.6513	4.6513	1
10	Cherries, packed in syrup or water ...	4.5257	4.5257	4.5257	1

- The result is ordered by the average price in descending order, so you will see grocery items with higher average retail prices first. So, this allows to analyze the priceobservations for different grocery items, helping to identify trends and patterns in the data.

What challenges and opportunities exist in enhancing access to affordable and nutritious fruits and vegetables?

```
SELECT
    Groceries.Item,
    AVG(prices.RetailPrice) AS AveragePrice,
    MIN(prices.RetailPrice) AS MinPrice,
    MAX(prices.RetailPrice) AS MaxPrice,
    AVG(prices.CupEquivalentPrice) AS AverageCupEquivalentPrice,
    MIN(prices.CupEquivalentPrice) AS MinCupEquivalentPrice,
    MAX(prices.CupEquivalentPrice) AS MaxCupEquivalentPrice
FROM Groceries
JOIN
    Prices  ON Groceries.UId = prices.UId
GROUP BY
    Groceries.Item
ORDER BY
```

```
    AveragePrice DESC;
```

	item character (64)	averageprice double precision	minprice double precision	maxprice double precision	averagecupequivalentprice double precision	mincupequivalentprice double precision	maxcupequivalentprice double precision
1	Figs	6.8371	6.8371	6.8371	1.1776	1.1776	1.1776
2	Mangoes	5.852	1.1513	10.5527	0.95585	0.5898	1.3219
3	Olives	5.7719	5.7719	5.7719	1.7179	1.7179	1.7179
4	Plum (prunes)	5.7042	5.7042	5.7042	1.0689	1.0689	1.0689
5	Dates	5.5713	5.5713	5.5713	0.9212	0.9212	0.9212
6	Raspberries	5.4134	4.1877	6.6391	1.7978	1.3849	2.2107
7	Blackberries	4.8267	3.6362	6.0172	1.6031	1.2025	2.0037
8	Apricots	4.79265	2.9665	6.6188	1.0544	0.9485	1.1603
9	Cranberries	4.6513	4.6513	4.6513	0.5729	0.5729	0.5729
10	Cherries, packed in syrup or water ...	4.5257	4.5257	4.5257	3.07	3.07	3.07

- From this result we can observe the pricing patterns of different grocery items, understanding the variation in retail and cup equivalent prices, and identifying items that may be more affordable or expensive on average. It provides insights into the cost distribution for various fruits and vegetables.

Find the percentage of total retail value contributed by each item within its form

```
SELECT
  Groceries.Item,
  Groceries.Form,
  prices.RetailPrice,
  (SUM(Groceries.Yield * prices.RetailPrice) OVER (PARTITION BY
Groceries.Form) / SUM(Groceries.Yield * prices.RetailPrice) OVER ()) *
100 AS PercentOfTotalRetailValue
FROM Groceries
JOIN prices  ON Groceries.UId = prices.UId;
```

	item character (64)	form character (64)	retailprice double precision	percentoftotalretailvalue double precision
1	Pumpkin	Canned ...	2.0172	14.2010006122104
2	Artichoke	Canned ...	3.4119	14.2010006122104
3	Mustard greens	Canned ...	1.0496	14.2010006122104
4	Mixed vegetables, peas & carrots ...	Canned ...	1.5339	14.2010006122104
5	Black beans	Canned ...	1.0281	14.2010006122104
6	Pineapple, packed in juice	Canned ...	1.4344	14.2010006122104
7	Peaches, packed in syrup or water ...	Canned ...	1.8117	14.2010006122104
8	Apricots, packed in juice	Canned ...	1.6905	14.2010006122104
9	Potatoes	Canned ...	1.107	14.2010006122104
10	Blackeye peas	Canned ...	1.0375	14.2010006122104

- From the output we can understand the relative contribution of each item to the overall retail value, considering the different forms of grocery items.

Within each form category, arrange items based on their cup equivalent prices.

```
SELECT
  Groceries.Item,
  Groceries.Form,
  prices.CupEquivalentPrice,
  RANK() OVER (PARTITION BY Groceries.Form ORDER BY
prices.CupEquivalentPrice) AS CupEquivalentPriceRank
FROM Groceries
JOIN Prices  ON Groceries.UId = prices.UId;
```

	item character (64)	form character (64)	cupequivalentprice double precision	cupequivalentpricerank bigint
1	Green beans	Canned	0.4659	1
2	Pinto beans	Canned	0.5189	2
3	Mustard greens	Canned	0.534	3
4	Tomatoes	Canned	0.5496	4
5	Carrots	Canned	0.5565	5
6	Turnip greens	Canned	0.566	6
7	Corn	Canned	0.5757	7
8	Apples, applesauce	Canned	0.5758	8
9	Kidney beans	Canned	0.5768	9
10	Potatoes	Canned	0.582	10

- From the output we can observe that the ranking is ascending, so the record with the lowest cup equivalent price within each form will have a rank of 1, the second-lowest will have a rank of 2, and so on.

Final Conclusions

In conclusion, our project “Grocery Price Analysis”, employed SQL queries to analyze a food dataset, the well-crafted database schema demonstrated effective data retrieval capabilities. We worked on DML, DDL statements to gain insights interesting facts from the dataset.