# Tarea 1 - Esquemas y Gráficos

**IIC2440 Procesamiento de Datos Masivos**

Maria Jose Ortega Rosales y Pedro Pablo Zavala Tejos

## 1. Creando un modelo de datos y sus índices

### 1.1 Modelo relacional e índices

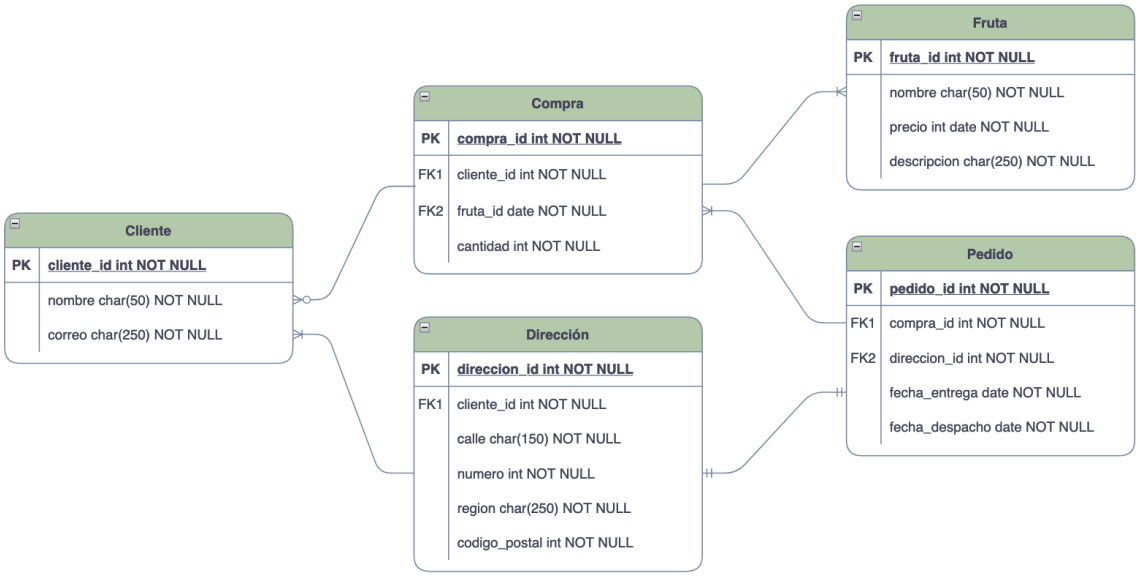


Figura 1. Modelo relacional de la base de datos.

En la figura 1, podemos visualizar diferentes tablas para el modelo de datos de la aplicación. Por un lado, tenemos la tabla **Cliente**, que tiene como llave primaria el atributo `cliente_id`, y también presenta otros atributos importantes, como el nombre y correo electrónico de cada cliente. Por otro lado, tenemos la tabla **Fruta**, cuya llave primaria es `fruta_id`, también tiene otros atributos principales, como el nombre de la fruta, su precio (unitario) y la descripción. Como cada cliente puede realizar 0 o muchas compras, y cada una de ellas puede tener al menos una fruta distinta, hemos diseñado la tabla **Compra**. Esta tabla presenta como llave primaria `compra_id`, y llaves foráneas `fruta_id` e `cliente_id`, apuntando a las llaves primarias de la tabla Fruta y Cliente, respectivamente. Además, tiene el atributo cantidad, cuyo valor es a un entero que indica la cantidad comprada de una fruta determinada.

Por otra parte, cada cliente al momento de realizar una compra, puede solicitar la entrega en una de sus direcciones. Por consiguiente, hemos considerado que cada cliente debe contar con al menos una dirección. Por lo tanto, creamos una nueva tabla **Dirección**, cuya llave primaria es `direccion_id`, y presenta como llave foránea `cliente_id`. Además, contiene algunos atributos característicos como: el número de la calle del cliente; el nombre de la calle (definido como calle); la región donde se encuentra, y el código postal.

Finalmente, creamos una tabla **Pedido**, que agrupará cada compra realizada por un cliente con alguna de sus direcciones. Además, contiene otros atributos, como la fecha de entrega y despacho del pedido. En esta tabla tenemos como llave primaria `pedido_id`, y como llaves foráneas `compra_id`, y `direccion_id`.

En específico, crearemos los siguientes índices para cada tabla:

1. **Cliente**: índice *clustered* `cliente_id` como llave primaria.
2. **Fruta**: índice *clustered* `fruta_id` como llave primaria.
3. **Compra**: índice *clustered* `compra_id` como llave primaria, e índices *unclustered* `cliente_id` y `fruta_id`.
4. **Dirección**: indice *clustered* `direccion_id` como llave primaria e índice *unclunstered* `cliente_id`.
5. **Pedido**: índice *clustered* `pedido_id` como llave primaria e índices *unclustered* `compra_id` y `direccion_id`

## 1.2. Planes de consultas

### 1.2.1. Consulta 1: "Entrega la fruta con identificador *i* junto a toda su información."

```
SELECT *
FROM Fruta
WHERE fruta_id = i
```

Figura 2. Consulta 1 en SQL

En esta consulta, utilizamos la llave primaria `fruta_id` para acceder directamente a la información de la i-ésima tupla de la tabla Fruta. La creación de un índice en la tabla nos permite solamente acceder a una información determinada en un cómputo constante, y no tener que iterar todas las tuplas.

### 1.2.2. Consulta 2: "Para un usuario, entrega todos sus pedidos, junto con la dirección, fecha de despacho, fecha de entrega y total en pesos ($) del pedido."

```
SELECT pedido_id, calle, numero, fecha_despacho,
      fecha_entrega, SUM(Compra.cantidad * Fruta.precio) AS cantidad_total
FROM Direccion, Pedido, Compra, Fruta
WHERE Direccion.direccion_id = Pedido.direccion_id
  AND Pedido.compra_id = Compra.compra_id
  AND Fruta.fruta_id = Compra.fruta_id
  AND Direccion.cliente_id = ClienteID
GROUP BY Pedido.pedido_id, calle, numero, fecha_despacho, fecha_entrega
```

Figura 3. Consulta 2 en SQL

En esta consulta hacemos un *Join* entre las tablas Dirección, Pedido, Compra y Fruta para poder acceder a toda la información relacionada a un cliente determinado y su compra. En primer lugar, si realizamos "`AND Direccion.cliente_id = ClienteID`", podemos acceder al índice que identifica a este cliente (con `cliente_id`) en la tabla Dirección. Con esta operación, solo obtenemos las tuplas necesarias de sus direcciones, ya que el índice `cliente_id` apunta solamente a la tupla con información del cliente `ClienteID`.

En segundo lugar, realizamos "`WHERE Direccion.direccion_id = Pedido.direccion_id`". Ya que, solo obtenemos los pedidos identificados con la llave foránea `direccion_id` para algunas direcciones del cliente `ClienteID`. Esto nos permite tener una consulta más eficiente, dado que no tendríamos que iterar por cada tupla de la tabla Dirección.

Finalmente, realizamos "`AND Pedido.compra_id = Compra.compra_id`" para obtener las compras relacionadas al cliente `ClienteID`, donde relacionamos cada llave `compra_id` con respecto a las llaves primarias de Compra. Esto nos permite filtrar solamente las tuplas relacionadas a las compras del cliente `ClienteID`. Además, para calcular el total de pesos del pedido, accedemos al precio de cada fruta con el índice `fruta_id`.

En conclusión, al hacer un *index nested loop join* nos permite minimizar el costo de las consultas, ya que solo obtenemos información relacionada a un cliente determinado. Esto permite no tener que iterar toda la tabla Cliente.

### 1.2.4. Consulta 3: "Para una compra de un usuario, entrega cada fruta junto a la cantidad que se compró, y el total en pesos ($) gastado en la compra separado por fruta."

```
SELECT fruta_id,
       SUM(cantidad) AS cantidad_total,
       SUM(cantidad * Fruta.precio) AS gasto_total
FROM Fruta, Compra, Pedido
WHERE Pedido.compra_id = Compra.compra_id
  AND Compra.fruta_id = Fruta.fruta_id
  AND Compra.cliente_id = ClienteID
  AND compra_id = CompraID
GROUP BY fruta_id
```

Figura 4. Consulta 3 en SQL

En esta consulta, podemos acceder a una compra de un usuario determinado mediante los índices `compra_id` y `cliente_id`, lo que nos permite evitar iterar por ambas tablas, realizando un *index nested loop join* envés de un *nested loop join.* Por lo tanto, en la tabla Pedido, obtenemos todas las compras de un cliente con su índice, y luego determinamos una compra en específico, indexando con un valor `CompraID`. Después, se realiza un *index nested loop join* entre las tablas Compra y Fruta, para obtener toda la información con respecto al gasto total y cantidad de cada fruta comparada por separado.

Tal como mencionamos anteriormente, la indexación nos permite acceder en un costo constante a cada tupla de una tabla, como por ejemplo, al acceder a una compra o fruta determinada.

### 1.2.4. Consulta 4: "Para una fruta en particular, entrega los usuarios que la compraron en un mes *m*."

```
SELECT cliente_id
FROM Compra
JOIN Pedido ON Compra.compra_id = Pedido.compra_id
WHERE EXTRACT(MONTH FROM Pedido.fecha_despacho) = m
AND Compra.fruta_id = FrutaID
GROUP BY Compra.cliente_id
```

Figura 5. Consulta 4 en SQL

En esta consulta, asumimos que la fecha de compra es igual a la fecha de despacho. En este caso, realizamos un *index nested loop join* entre las tablas Compra y Pedido. Después, utilizamos `fruta_id` para identificar una fruta en particular, lo que nos retorna todas las tuplas relacionadas a esta fruta. Y finalmente, establecemos la condición "`WHERE EXTRACT(MONTH FROM Pedido.fecha_despacho) = m`", la cual nos permite filtrar solamente las compras realizadas en un mes *m*.

Por un lado, esta consulta es medianamente eficiente. Si bien, realizamos una indexación con una llave primaria `Compra.compra_id`, y accedemos a toda información relacionada la fruta `FrutaID` en tiempo constante. Sin embargo, estamos accediendo a tuplas con un atributo que no fue indexado, que es el caso de `fecha_despacho`. La consulta actual no responderá de una manera tan eficiente, ya que estamos indexando con un atributo que se repite en más de una tupla – ya que existen otras compras con la misma fecha de despacho –.

# 2. Analítica en un Data Warehouse

## 2.1

Entiendo que mi amigo necesitará tomar decisiones empresariales dado ciertos análisis, por lo necesita de información que se presente de forma eficiente, que sea más precisa y detallada y correr consultas directamente en la base de datos de la aplicación afectaría al rendimiento del sistema, especialmente si la base de datos está siendo utilizada activamente para otras operaciones de la aplicación lo que podría causar por ejemplo la interrupción del servicio para los usuarios, por lo que cargar los datos en un Data Warehouse es preferible por varias razones, partiendo por que están optimizados para consultas analíticas complejas sobre conjuntos de datos grandes (como podría llegar a ser los de una empresa)dando tiempos de respuesta rápidos incluso para consultas complejas, a diferencia de las bases de datos de aplicaciones que suelen estar diseñadas para manejar transacciones en tiempo real y no están optimizadas para consultas analíticas más complejas sobre grandes volúmenes de datos. También mantienen un historial completo de los datos a lo largo del tiempo, lo que es esencial para el análisis de tendencias y/o la generación de informes comparativos a lo largo de diferentes períodos, como los que se necesitarán en el área de ventas. Otra razón, es que permite integrar datos de múltiples fuentes y formatos en un único repositorio centralizado,esto facilita la combinación y análisis de datos de diferentes áreas de la empresa.

## 2.2

El siguiente diagrama muestra las tablas principales normalizadas luego de procesar los datos *json*.

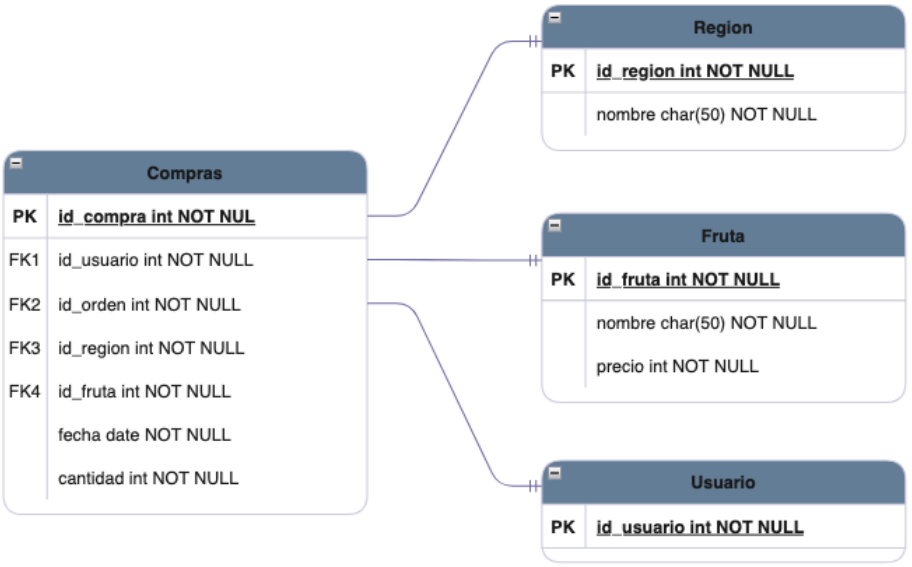


Figura 2. Modelo relacional de la base de datos resultante.

Consideramos solamente las 4 tablas de la figura 2, ya que son las que trabajaremos principalmente en las consultas. Sin embargo, también podríamos considerar otras tablas relación entre entidades, como RegionUsuario, y RegionFruta, pero para efectos de la tarea, solo creamos archivos para las tablas anteriores. En contexto al archivo json, podríamos decir que un usuario podría comprar en más de 1 región, y cada orden podría contener al menos 1 fruta. Por otra parte, en cada región compran al menos 1 usuario. La tabla Compras es considerada como una tabla relación entre las tablas Region, Fruta y Usuario.

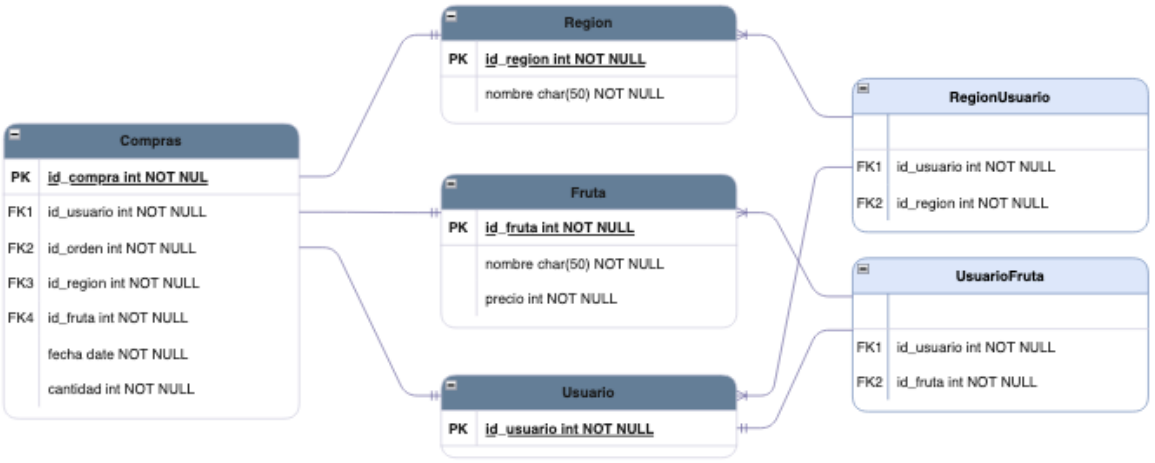


Figura 3. Modelo relacional de la base de datos completo.

## 2.4. Gráficos

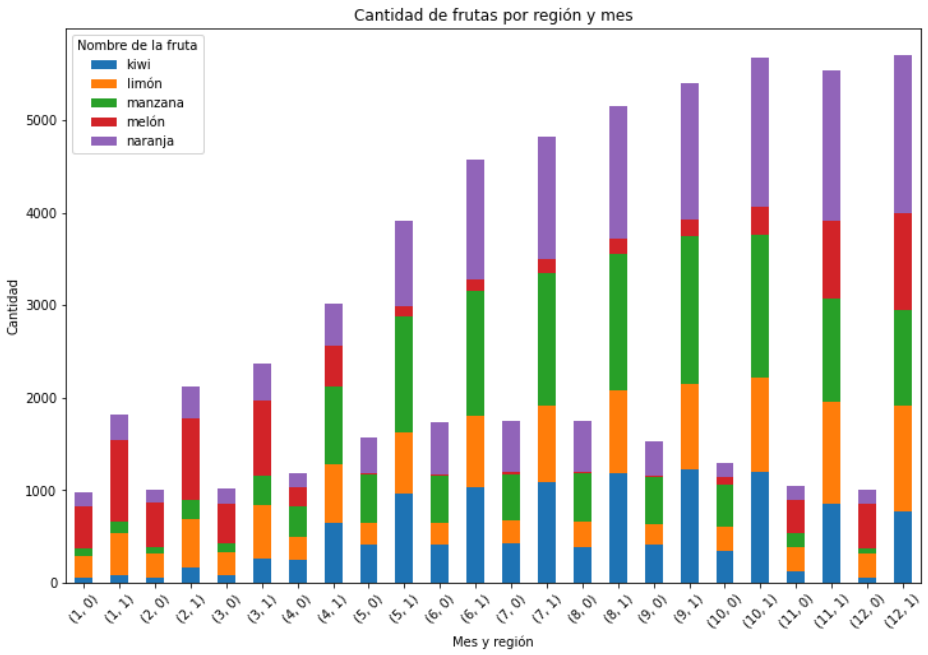


Gráfico 1. Cantidad de frutas por región y mes

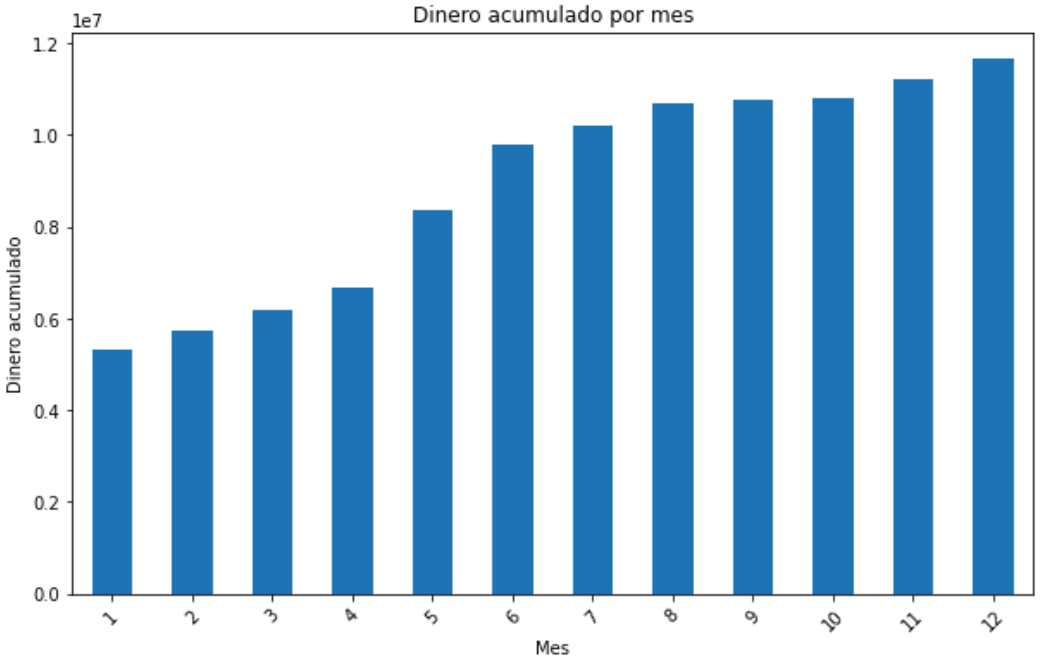


Gráfico 2. Dinero que ha entrado a la tienda

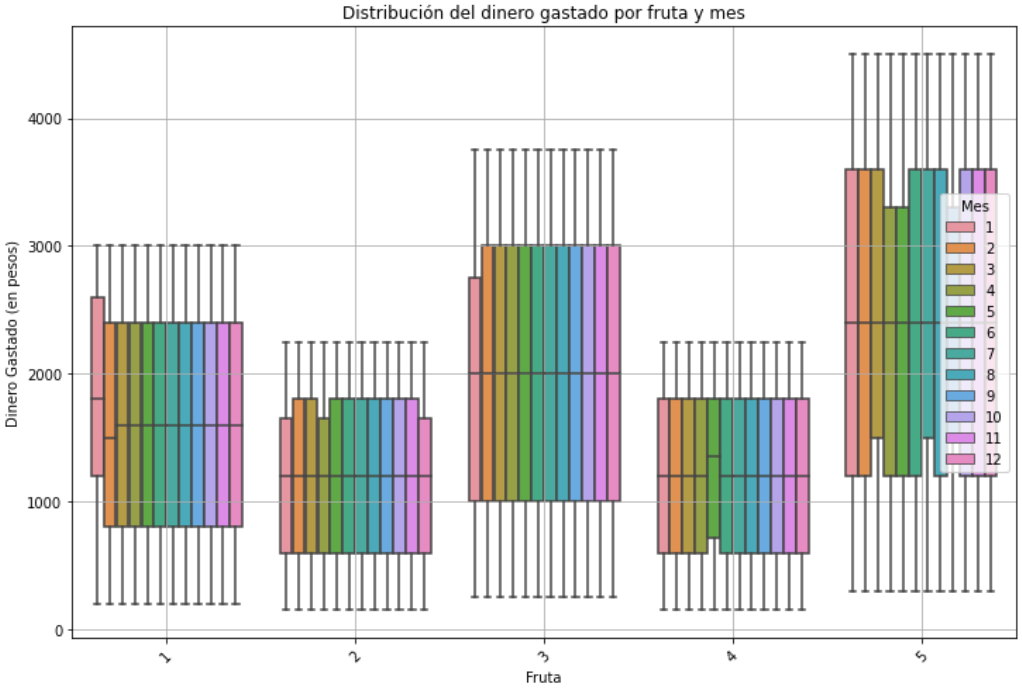


Gráfico 3. La distribución del dinero que gastan los clientes en la tienda

### 2.4.1 Análisis de los gráficos

**a. ¿Qué región crees que va en crecimiento y va a aumentar su demanda el siguiente año?**

La región 2 según el gráfico 1 se nos muestra como notoriamente presenta más ventas que la región 1, especialmente a medida que pasa el tiempo, por lo que podemos inferir que va en crecimiento, además la naranja es la más demandada a lo largo del tiempo.

**b. ¿Puedes ver alguna relación entre la venta de las frutas?**

- Comparación entre regiones:

  En general, las ventas de frutas en la región 2 son considerablemente más altas que en la región 1 en la mayoría de meses.

- Tendencias estacionales:

  Se observa un aumento en las ventas de frutas en ambas regiones durante los meses de invierno - otoño, lo que sugiere una mayor demanda de la fruta en esa época del año, especialmente en las naranjas, y esto tiene que ver con los meses en que empieza la temporada de esta fruta .

  En contraste, los meses de calor después de invierno muestran ventas más bajas en la región 1, a diferencia de la región que casi no presenta baja de ventas en el segundo semestre, pero si se disminuye un poco la venta de manzanas

- Comparación de ventas de frutas:

  La naranja es consistentemente las más vendida en ambas regiones a lo largo del año.

- Identificación de picos y valles de ventas:

  Se observan picos de ventas en las dos regiones durante los meses de octubre y diciembre para la región 2, y para la region 1 de mayo a octubre.

  Los meses de mayo y diciembre muestran los valles más bajos de ventas en la región 1 , mientras que para la región 2, este es en enero y marzo.

**c. Tu amigo tiene una predicción de la demanda del kiwi para el próximo año en distintas regiones. ¿Cómo lo harías para predecir la venta de una o más frutas en base a esa predicción?**

Para predecir la demanda de una o más frutas en base a la predicción de mi amigo para el próximo año, utilizaría un enfoque de modelado predictivo. Por lo que recopilar los datos históricos de ventas de las frutas de interés, en este caso el kiwi, en las distintas regiones durante un período de tiempo largo, datos que serían para entrenar un modelo predictivo.

Como modelo,optaría por utilizar un modelo de regresión lineal debido a su capacidad para modelar relaciones lineales entre variables. En este caso, la variable independiente serían los datos históricos de ventas de frutas por mes, y la variable dependiente sería la predicción de la demanda de kiwi o de otras frutas para el próximo año.

Una vez que se ha recopilado y limpiado adecuadamente el conjunto de datos, los dividimos en conjuntos de entrenamiento y prueba. El conjunto de entrenamiento se utilizaría para ajustar los coeficientes del modelo de regresión lineal, mientras que el conjunto de prueba se utilizaría para

evaluar la precisión y la generalización del modelo, luego se valida el modelo utilizando métricas de evaluación para garantizar que la predicción sea lo más precisa posible.

Además, se podrían incluir otros factores (se necesitaría de otra investigación) que podrían influir en la demanda de frutas, como el clima, la estacionalidad, las tendencias del mercado, entre otros. Estos factores podrían integrarse en el modelo como variables adicionales para mejorar su precisión y capacidad predictiva.