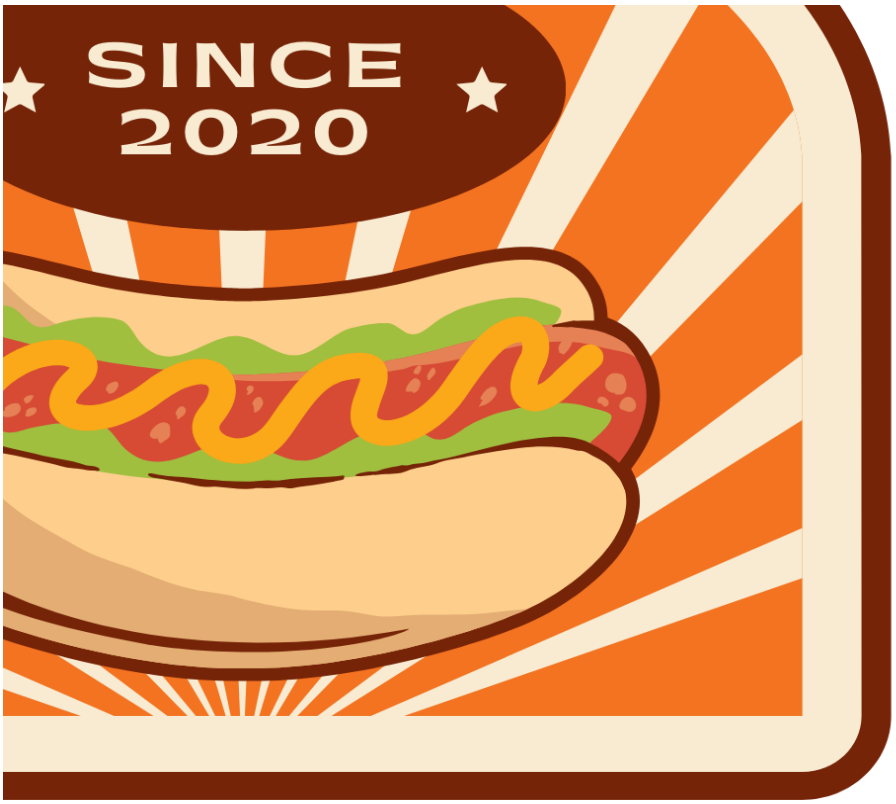


PROYECTO
SQL

Descubriendo la Base de Datos FastFood

Presentado por

LUCAS GEBHARDT

HENRY

Carrera: Data Analytics

Módulo 2

Descubriendo la Base de Datos FastFood

Nombre del autor: Lucas Damián Gebhardt

Email: lucasdamiangebhardt@gmail.com

Cohorte: DA-FT01

Fecha de entrega: 21/03/2024

Institución: Henry FastFood: una tienda de comidas rápidas que cuenta con distintas sucursales y ofrece un menú variado a lo largo del día.

Introducción

El proyecto consistió en la creación de una base de datos para la empresa ***Henry FastFood***, una cadena ficticia de restaurantes de comida rápida.
El objetivo principal se centró en el diseño un sistema de gestión de datos, mediante la implementación de lenguaje SQL, que permitiera registrar y analizar operaciones comerciales de la empresa.
Cabe destacar, que a pesar de su gran éxito con aperturas de nuevas sucursales, tenían una gestión de información dispersa en archivos de Google Sheets y documentos manuales.
Para lograr esto, fue necesario la implementación de los sublenguajes DDL y DML para la creación y agregados de datos respectivamente, como así también la implementación de funciones, vistas, procedimientos almacenados y sentencias como CASE, JOINs, etc. que veremos a lo largo el desarrollo de este informe.
Lograr una implementación eficiente de las soluciones a los requerimientos de la compañía, nos asegura una base de datos consolidada, la cual se pueda escalar y en la cual se pueda extraer datos de manera eficiente para la toma de decisiones.
Para llevar adelante el proyecto, se utilizó SQL Server Management Studio, herramienta que permite la gestión y administración de los componentes de la base de datos

Carrera: Data Analytics

Módulo 2

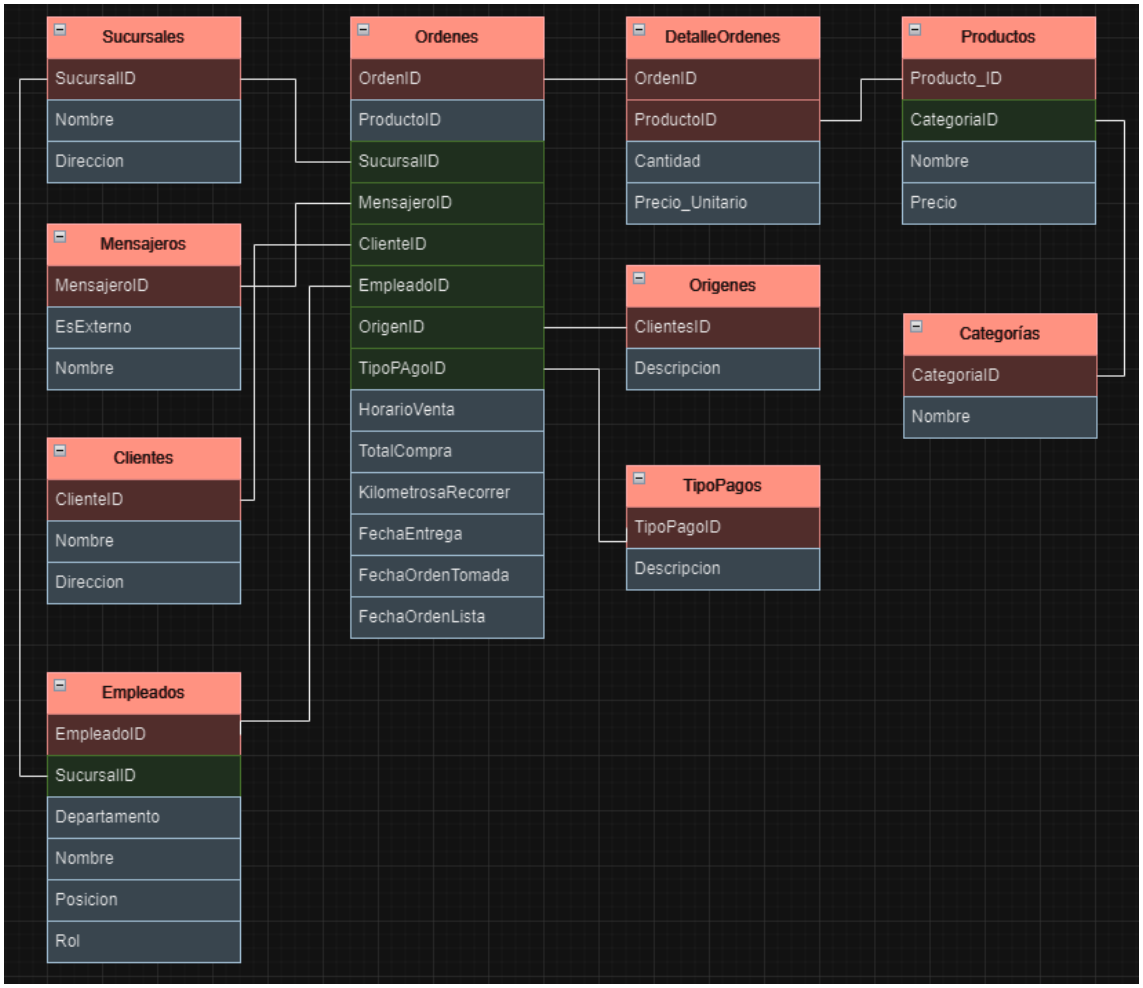
Desarrollo del proyecto

El primer paso, fue entender los requerimientos que Henry FastFood necesitaba para su negocio. Es importante tener en claro varios aspectos que son particulares de cada compañía. En Henry FastFood como no había ningún sistema organizado, se identificaron las entidades que iban a formar parte del sistema y las relaciones que iban existir entre ellas.

Una vez identificadas las entidades y ya con el modelo semántico desarrollado, fue necesario como parte de las buenas prácticas, la creación de un modelo de entidad relación, el cual a futuro servirá como base para cuando se creen las tablas y relaciones dentro de la base de datos.

En la siguiente imagen podemos verlo en detalle.

Para esta parte del proyecto, se utilizó la herramienta Draw.io que nos permite un maquetado rápido y visualmente atractivo.



Carrera: Data Analytics

Módulo 2

Avance N1:

Empleando el Lenguaje de Definición de Datos (DDL) se realizó la creación de las tablas, definición de los tipos de campos y se establecieron las relaciones entre tablas.
Para el desarrollo en SQL Server, se optó por utilizar la notación CamelCase para darle mayor legibilidad y estandarización a los elementos de la base de datos. Otra decisión, fue crear primero las tablas y una vez creadas todas, generar las relaciones entre sí, para evitar confusiones e inconsistencias al momento de crearla.
Las Queries correspondientes a este avance, se pueden ver como archivo anexo a este informe bajo el nombre *DA_Gebhardt_Lucas_avance1PI.sql.* Estos códigos al ejecutarse crean en primera instancia la Base de Datos, luego utiliza la sentencia USE para poder posicionarse sobre la misma y poder trabajar en ella, y luego por medio de la sentencia CREATE se crean las tablas necesarias.

A modo de ejemplo, a continuación, se muestra el código de creación de una de las tablas:

```
CREATE TABLE Productos(
	ProductoID INT PRIMARY KEY IDENTITY ,
	Nombre VARCHAR(255),
	CategoriaID INT, -- FK, referencia a PK de la tabla Categoria
	Precio DECIMAL(10,2) NOT NULL DEFAULT (0)
);
```

Como se mostró en el modelo de entidad relación, es necesario que ciertas tablas se relacionen entre si para asegurar la integridad del sistema. Es por eso, que luego de crear todas las tablas necesarias, se generaron las relaciones a través de las Foreign Keys y Primary Keys.
A continuación, se muestra un ejemplo de estas sentencias, el resto puede verse en el archivo anexo correspondiente:

```
ALTER TABLE Ordenes

ADD CONSTRAINT FK_Ordenes_Sucursales
FOREIGN KEY(SucursalID) REFERENCES Sucursales(SucursalID);
```

Una vez generadas todas las relaciones, se pueden hacer las inserciones de datos correspondientes. Esto se detalla a continuación en el Avance N2.

Carrera: Data Analytics

Módulo 2

Avance N2:

Ya con la estructura creada de la base de datos, podemos comenzar a utilizar sentencias propias del Lenguaje de Modelado de Datos (DML) para el poblado de todas las tablas.
Se insertó mediante la sentencia INSERT la totalidad de registros de cada tabla. Estos datos fueron brindados por Henry, y consta de 10 registros para cada tabla.
Luego, en forma de ejemplos se realizaron UPDATEs y DELETEs.

Al igual que el caso anterior, las queries correspondientes a este avance se encuentra como anexo bajo el nombre de *DA_Gebhardt_Lucas_avance2PI.sql.*
Un ejemplo de inserción de datos puede verse a continuación:

```
-- Insertar datos en Productos
INSERT INTO Productos (Nombre, CategoriaID, Precio)
VALUES
	('Hamburguesa Deluxe', 1, 9.99),
	('Cheeseburger', 1, 7.99),
	('Pizza Margarita', 10, 11.99),
	('Pizza Pepperoni', 10, 12.99),
	('Helado de Chocolate', 7, 2.99),
	('Helado de Vainilla', 7, 2.99),
	('Ensalada César', 4, 5.99),
	('Ensalada Griega', 4, 6.99),
	('Pastel de Zanahoria', 2, 3.99),
	('Brownie', 2, 2.99);
```

Un ejemplo de modificación de datos con UPDATE:

```
-- Cambiar la posición de todos los empleados del departamento 'Cocina' a 'Chef'
UPDATE Empleados
SET Posicion = 'Chef'
WHERE Departamento = 'Cocina';
```

En cuanto a los DELETEs, para no romper la integridad de las tablas a trabajar se realizaron en una base de datos de prueba.

En esta instancia, se lograron responder preguntas mediante el armado de queries como la que se ve a continuación:

```
-- 5 ¿Cuáles son las sucursales con un promedio de ventas por orden superior a un
valor específico, ordenadas por el promedio de kilómetros recorridos para las
entregas de mayor a menor? --

SELECT SucursalID,
		CAST(AVG(TotalCompra) AS DECIMAL(10,2)) AS [Promedio Venta Por
Orden],
		CAST(AVG(KilometrosRecorrer) AS DECIMAL(10,2)) AS [Promedio de KM]
FROM Ordenes
GROUP BY SucursalID,KilometrosRecorrer
HAVING AVG(TotalCompra) > 50.00
ORDER BY KilometrosRecorrer DESC;
```

Carrera: Data Analytics

Módulo 2

Avance N3:

En esta tercera instancia, se necesitan resolver preguntas específicas que requieren consultas más avanzadas.
Algunos ejemplos son:

```
-- 9. ¿Cómo se comparan las ventas promedio antes y después del 1 de julio de
2023?

SELECT CAST(AVG(TotalCompra) AS DECIMAL(10,2)) AS [Venta Promedio],
       'Promedio Venta Antes de Julio 2023' AS Comentario
FROM Ordenes
WHERE FechaOrdenTomada < '2023-07-01'
UNION -- Anexo las cosultas individuales
SELECT CAST(AVG(TotalCompra) AS DECIMAL(10,2)) AS [Venta Promedio],
       'Promedio Venta Despues de Julio 2023' AS Comentario
FROM Ordenes
WHERE FechaOrdenTomada > '2023-07-01';


-- 10. Pregunta: ¿Durante qué horario del día (mañana, tarde, noche) se registra
la mayor cantidad de ventas, ¿cuál es el valor promedio de estas ventas, y cuál
ha sido la venta máxima alcanzada?

SELECT
    HorarioVenta,
    COUNT(OrdenID) AS CantidadVentas,
    CAST(AVG(TotalCompra) AS DECIMAL(10,2)) AS VentaPromedio,
    MAX(TotalCompra) AS VentaMaxima
FROM
    Ordenes
GROUP BY
    HorarioVenta
ORDER BY
    VentaMaxima DESC; -- Se ordena por venta máxima en este caso para
ejemplificar, ya que si ordenamos por cantidadVentas en este caso los 3 turnos
tienen 3 ventas y no veríamos un orden.
```

Para ver el resto de las consultas realizadas, se puede acceder al archivo *DA_Gebhardt_Lucas_avance3PI.sql* anexo a este informe.

Carrera: Data Analytics

Módulo 2

Avance N4:

En este último avance, se trabaja con múltiples tablas en una sola consulta. Esto se puede llevar a cabo mediante las sentencias JOINs, subconsultas y el correcto uso de alias para dar orden y legibilidad a la query.

Se plantean 5 preguntas de negocio, cuyas respuestas pueden encontrarse en el archivo *DA_Gebhardt_Lucas_avance4PI.sql* anexo al informe.
A modo de ejemplo , se muestra la querie para responder una de ellas y la tabla resultante con lo respuesta

```
/*4. Detalle completo de órdenes - ¿Cómo puedo obtener un detalle completo de las
órdenes, incluyendo cliente, empleado que tomó la orden, y el mensajero que la
entregó?*/

SELECT o.ordenID,
       o.HorarioVenta,
       o.TotalCompra,
       o.KilometrosRecorrer,
       o.FechaOrdenLista,
       c.Nombre AS [Nombre del cliente],
       c.Direccion AS [Direccion del cliente],
       e.Nombre As [Orden Tomada Por],
       m.Nombre AS [Orden Entregada Por]
FROM Ordenes o
INNER JOIN
       Clientes c ON o.ClienteID = c.ClienteID
INNER JOIN
       Empleados e ON o.EmpleadoID = e.EmpleadoID
INNER JOIN
       Mensajeros m ON o.MensajeroID = m.MensajeroID;
```

	ordenID	HorarioVenta	TotalCompra	KilometrosRecorrer	FechaOrdenLista	Nombre del cliente	Direccion del cliente	Orden Tomada Por	Orden Entregada Por
1	1	Mañana	50.00	5.50	2023-01-10 08:15:00.000	Cliente Uno	1000 A Street	John Doe	Mensajero Uno
2	2	Tarde	75.00	10.00	2023-02-15 14:00:00.000	Cliente Dos	1001 B Street	Jane Smith	Mensajero Dos
3	3	Noche	20.00	2.00	2023-03-20 19:15:00.000	Cliente Tres	1002 C Street	Bill Jones	Mensajero Tres
4	4	Mañana	30.00	0.50	2023-04-25 09:15:00.000	Cliente Cuatro	1003 D Street	Alice Johnson	Mensajero Cuatro
5	5	Tarde	55.00	8.00	2023-05-30 15:15:00.000	Cliente Cinco	1004 E Street	Tom Brown	Mensajero Cinco
6	6	Noche	45.00	12.50	2023-06-05 20:15:00.000	Cliente Seis	1005 F Street	Emma Davis	Mensajero Seis
7	7	Mañana	65.00	7.50	2023-07-10 08:15:00.000	Cliente Siete	1006 G Street	Lucas Miller	Mensajero Siete
8	8	Tarde	85.00	9.50	2023-08-15 14:15:00.000	Cliente Ocho	1007 H Street	Olivia García	Mensajero Ocho
9	9	Noche	95.00	3.00	2023-09-20 19:15:00.000	Cliente Nueve	1008 I Street	Ethan Martinez	Mensajero Nueve

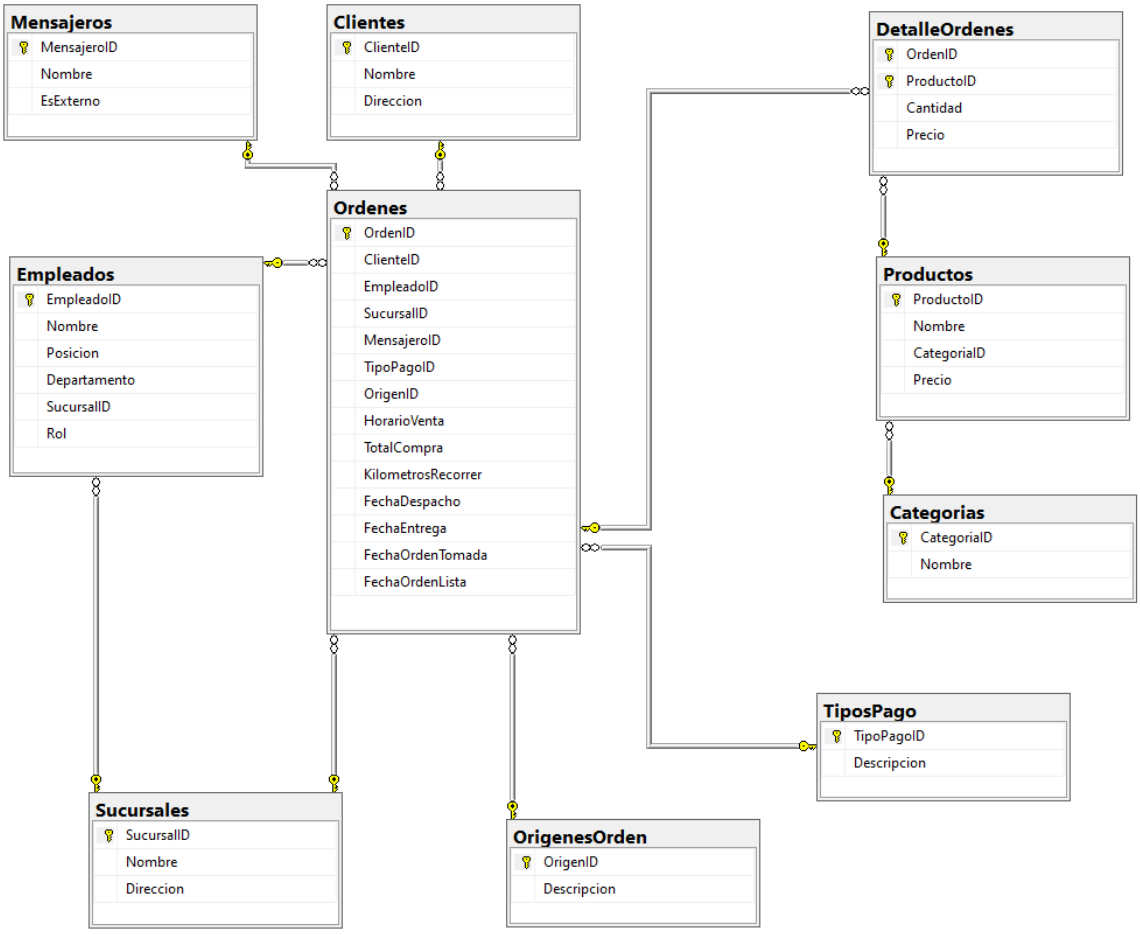
Carrera: Data Analytics

Módulo 2

Resultados y consultas

Una vez creado el sistema, la ventaja que ofrece SQL Server Management Studio es la posibilidad de crearnos un Diagrama de Entidad Relación en base a las tablas y las conexiones que generamos entre ellas en la primera etapa.

Como se ve en esta imagen, se respetan las entidades y relaciones con las cuales se comenzó el proyecto. El Mockup del diagrama hecho en Drawio, finalmente se representa de la misma manera en SQL Server:



Además de las consultas de negocios de los avances anteriores, se plantean 6 interrogantes los cuales deben responderse a través de la implementación de consultas avanzadas. En las próximas hojas se muestran cada una de las queries de consulta, los hallazgos claves obtenidos, y una serie de recomendaciones estratégicas basadas en el análisis de estos resultados.

Carrera: Data Analytics

Módulo 2

Consultas finales y hallazgos claves

Eficiencia de los mensajeros:

```
-- Cuál es el tiempo promedio desde el despacho hasta la entrega de los pedidos
por los mensajeros?

SELECT
    m.MensajeroID,
    m.Nombre,
    AVG(DATEDIFF(MINUTE, o.FechaDespacho, o.FechaEntrega)) AS
TiempoPromedioDespachoEntrega
FROM
    Ordenes o
RIGHT JOIN
    Mensajeros m ON o.MensajeroID = m.MensajeroID
WHERE
    o.FechaDespacho IS NOT NULL
    AND o.FechaEntrega IS NOT NULL
    AND m.MensajeroID IS NOT NULL
GROUP BY
    m.MensajeroID,
    m.Nombre
ORDER BY
    TiempoPromedioDespachoEntrega;
```

Esta Query nos devuelve:

	MensajeroID	Nombre	TiempoPromedioDespachoEntrega
1	1	Mensajero Uno	30
2	2	Mensajero Dos	30
3	3	Mensajero Tres	30
4	4	Mensajero Cuatro	30
5	5	Mensajero Cinco	30
6	6	Mensajero Seis	30
7	7	Mensajero Siete	30
8	8	Mensajero Ocho	30
9	9	Mensajero Nueve	30

En esta BD particularmente, como tiene pocos registros y no hay un gran historial que nos permita realizar un análisis profundo, poco podemos decir de los hallazgos obtenidos. Pero si podemos inferir que esta consulta nos permitiría evaluar el rendimiento y/o desempeño de cada mensajero a la hora de entregar una orden.

Carrera: Data Analytics

Módulo 2

Análisis de Ventas por Origen de Orden:

```
-- Qué canal de ventas genera más ingresos?

SELECT od.Descripcion, SUM (TotalCompra) AS [Total de Ventas]
FROM Ordenes o
INNER JOIN OrigenesOrden Od ON o.OrigenID = od.OrigenID
GROUP BY od.Descripcion
ORDER BY [Total de Ventas] DESC;
```

	Descripcion	Total de Ventas
1	Presencial	140.00
2	Drive Thru	125.00
3	Teléfono	105.00
4	En línea	95.00
5	App Móvil	55.00

Con esta consulta, podemos corroborar que canal de venta está teniendo mayor rentabilidad, con los registros cargados en nuestra base de datos, podemos asegurar que el canal de venta "Presencial" es el que mayores ventas le genera a Henry FastFood.

Productividad de los Empleados:

```
-- ¿Cuál es el volumen de ventas promedio gestionado por empleado?

SELECT e.Nombre, COUNT(OrdenID) AS [Volumen de venta] FROM Ordenes o
INNER JOIN Empleados e ON o.EmpleadoID = e.EmpleadoID
GROUP BY e.Nombre;
```

	Nombre	Volumen de venta
1	Alice Johnson	1
2	Bill Jones	1
3	Emma Davis	1
4	Ethan Martinez	1
5	Jane Smith	1
6	John Doe	1
7	Lucas Miller	1
8	Olivia García	1
9	Tom Brown	1

Esta Query nos permite ver de manera rápida que cantidad de ordenes ha generado cada empleado. En el caso puntual de esta BD, al momento cada empleado solo ha generado una orden.
Esto podría permitir ver que empleados tienen el mayor volumen de ventas, para a partir de ese insight tomar medidas para mejorar las ventas del resto de empleados.

Carrera: Data Analytics

Módulo 2

Análisis de Demanda por Horario y Día:

```
¿Cómo varía la demanda de productos a lo largo del día?
NOTA: Esta consulta no puede ser implementada sin una definición clara del
horario (mañana, tarde, noche) en la base de datos existente.
Asumiremos que HorarioVenta refleja esta información correctamente. */

Select o.HorarioVenta AS [Horario de Venta],
              SUM(d.cantidad) AS [Demanda de productos]
FROM Ordenes o
LEFT JOIN DetalleOrdenes d ON o.OrdenID = d.OrdenID
GROUP BY o.HorarioVenta;
```

	Horario de Venta	Demanda de productos
1	Mañana	32
2	Noche	NULL
3	Tarde	NULL

Con esta consulta, se obtiene la demanda de productos de acuerdo con el horario de la venta. En este caso, las ventas que vemos son solo de la mañana (Ya que en la tabla detalle de venta solo se encuentra el detalle de la orden con OrdenID=1). Pero al momento de escalar esta base de datos, esto permitirá ver en qué momento del día hay mas flujo de consumo.

Comparación de Ventas Mensuales:

```
-- ¿Cómo se comparan las ventas mensuales de este año con el año anterior?

SELECT SUM(totalCompra) AS [Total de Ventas],
       MONTH(FechaOrdenTomada) AS [Mes],
       YEAR(FechaOrdenTomada) AS [Año]
FROM ordenes
GROUP BY FechaOrdenTomada;
```

	Total de Ventas	Mes	Año
1	50.00	1	2023
2	75.00	2	2023
3	20.00	3	2023
4	30.00	4	2023
5	55.00	5	2023
6	45.00	6	2023
7	65.00	7	2023
8	85.00	8	2023
9	95.00	9	2023

Con esta consulta estamos obteniendo las ventas mensuales por año, pero como todas las ordenes de esta base de datos son del 2023, no podemos ver otros registros en los cuales hacer la comparativa.
podría ser de utilidad organizarlos de forma descendente según total de ventas, para ver en las primeras filas aquellos meses con más ventas.

Carrera: Data Analytics

Módulo 2

Análisis de Fidelidad del Cliente:

```
¿Qué porcentaje de clientes son recurrentes versus nuevos clientes cada mes?
NOTA: La consulta se enfocaría en la frecuencia de órdenes por cliente para
inferir la fidelidad.*/
```

```
SELECT CLienteID,
          COUNT(OrdenID) AS [Número de Ordenes]
FROM Ordenes
GROUP BY ClienteID;
```

	CLienteID	Número de Ordenes
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	6	1
7	7	1
8	8	1
9	9	1

Con este código solo podríamos obtener la cantidad de ordenes respecto a cada cliente. Al ser acotada la BD, solo obtenemos una orden por cliente.
No hay información histórica para obtener la fidelidad del cliente.

```
-- Con este código más avanzado en cambio, podríamos ver reflejado en una tabla
los requerimientos pedidos.
WITH ClientesPorMes AS (
SELECT
    ClienteID,
    YEAR(FechaOrdenTomada) AS [Año],
    MONTH(FechaOrdenTomada) AS [Mes],
    COUNT(DISTINCT OrdenID) AS [NumeroOrdenes]
    FROM
        Ordenes
    GROUP BY
        ClienteID, YEAR(FechaOrdenTomada), MONTH(FechaOrdenTomada)
),
FidelidadCliente AS (
    SELECT
        Año,
        Mes,
        SUM(CASE WHEN NumeroOrdenes > 1 THEN 1 ELSE 0 END) AS Recurrentes,
        SUM(CASE WHEN NumeroOrdenes = 1 THEN 1 ELSE 0 END) AS Nuevos,
        COUNT(*) AS TotalClientes
    FROM
        ClientesPorMes
    GROUP BY
        Año, Mes
)
SELECT Año, Mes, Recurrentes, Nuevos, TotalClientes,
    CAST(Recurrentes AS FLOAT) / TotalClientes AS PorcentajeRecurrentes,
    CAST(Nuevos AS FLOAT) / TotalClientes AS PorcentajeNuevos
FROM
    FidelidadCliente
ORDER BY
    Año, Mes;
```

	Año	Mes	Recurrentes	Nuevos	TotalClientes	PorcentajeRecurrentes	PorcentajeNuevos
1	2023	1	0	1	1	0	1
2	2023	2	0	1	1	0	1
3	2023	3	0	1	1	0	1
4	2023	4	0	1	1	0	1
5	2023	5	0	1	1	0	1
6	2023	6	0	1	1	0	1
7	2023	7	0	1	1	0	1
8	2023	8	0	1	1	0	1
9	2023	9	0	1	1	0	1

Carrera: Data Analytics

Módulo 2

Recomendaciones estratégicas

A partir de los resultados obtenidos de las consultas anteriores, como analista de datos las recomendaciones estratégicas que se pueden dar a Henry FastFood son las siguientes:

- Reconocer y premiar el buen desempeño de aquellos mensajeros con mejor eficiencia a la hora de entregar la orden, con el fin de incentivar la eficiencia en la entrega de pedidos dentro del equipo de mensajeros.

- Fortalecer la promoción y el desarrollo de la aplicación móvil para impulsar las ventas y mejorar la experiencia del cliente.

- Reconocer y recompensar el desempeño destacado de aquellos empleados con mayor productividad para motivar al equipo y mejorar la productividad general.

- Ajustar el inventario y los recursos operativos para satisfacer la demanda en los horarios de mayor actividad. (en este caso en particular se vio que era la mañana, pero en base a una sola orden cargada en la base de datos)

- Crear algún tipo de beneficio extra para aquellos clientes recurrentes con mas de una cantidad razonable de órdenes.

Optimización y sostenibilidad

La optimización de la base de datos se ve desde el comienzo, desde la elección de la notación CamelCase, el mockup del DER, el armado propiamente de las tablas y relaciones.
Se armo el sistema de manera tal que los registros se guarden de manera optima respetando las buenas prácticas.
Por el lado de las consultas y queries, se evita abusar de la sentencia SELECT * FROM (“seleccionar todo de..”) y en cambio se puntualiza los datos necesarios que se necesita recoger. Con esto logramos que el motor de búsqueda trabaje de manera óptima, al tener pocos registros eso no es algo significativo, pero al momento de la expansión de la base de datos eso será una cuestión fundamental para que la base de datos funcione de manera correcta todo el tiempo.
Es importante señalar que en las queries de inserción de datos de datos provistas por Henry, se puede encontrar una pequeña inconsistencia entre la tabla Ordenes y la tabla DetalleOrdenes , ya que la columna Ordenes.TotalCompra debería traer la suma total de los valores de los productos correspondientes a un mismo número de orden (esto lo vemos en la columna DetalleOrdenes.OrdenID) y sin embargo, tienen un número que no está relacionado. Esto podría ser una mejora en la Base de Datos y optimización de la misma.
La creación de vistas y procedimientos almacenados también nos ayudará a un uso optimizado de las consultas.

Carrera: Data Analytics

Módulo 2

Desafíos y soluciones

Durante el desarrollo del proyecto, el principal desafío fue trabajar de manera analítica con tan poca cantidad de registros. Se tornaba difícil visualizar ciertas consultas, ya sean de las avanzadas o no.
La forma de solucionar ese inconveniente en principio fue crear una segunda base de datos auxiliar (clonada de la base de datos original) para realizar las pruebas necesarias y corroborar los scripts.
Otra solución a la falta de registros, fue modificar las especificaciones de requerimientos de las queries, por ejemplo, reemplazar un “WHEN X > 100” por un “WHEN x > 60”, ya que no había registros por encima del valor 100 en una determinada consulta y por lo tanto no se podía interpretar bien la tabla obtenida.

Reflexión personal

El proyecto integrador me proporcionó una visión detallada del funcionamiento y el rendimiento de Henry FastFood a través del análisis de datos y la utilización del lenguaje SQL.
Las consultas avanzadas van a permitir en el futuro identificar tendencias, patrones y oportunidades clave para la mejora continua del negocio.
Es clave como analista de datos poder destacar aspectos como la eficiencia de los mensajeros, el impacto de los distintos canales de ventas y la productividad del personal a través del manejo de una herramienta como SQL Server y poder “compartir” estos hallazgos fundamentales para la toma de decisiones estratégicas con el responsable del área del negocio al que estamos reportando.
En lo personal, el proyecto me sirvió para afianzar conocimientos que tenia vistos, pero que no había logrado comprender del todo (como los JOINs). Creo también, que la capacidad crítica se aumenta al pensar las queries y consultas que debemos implementar para darle una respuesta a las inquietudes del cliente.
Si tuviera que hacer este trabajo nuevamente, me aseguraría de sumar la mayor cantidad de entidades que sean representativas del negocio, para poder tener una mayor visión del sistema.

Carrera: Data Analytics

Módulo 2

EXTRA CREDIT

Para mejorar el sistema de gestión de datos de Henry FastFood, se pueden implementar diferentes medidas, siempre que respeten el criterio de escalabilidad y solidez con el que se trabajo durante todo el proyecto.

- En cuanto a la **expansión de datos**, si bien con las tablas creadas para este proyecto se pudo ver a priori un correcto análisis de datos de las ventas, puede ser sumamente importante tener también un conjunto de datos que hable del stock de insumos y que estos datos se relaciones, por ejemplo, con cada sucursal. Otra opción puede ser sumar conjuntos de datos referidos al marketing de la empresa, conjuntos con detalles financieros mas profundos que los que contiene la tabla detalleOrdenes, o conjuntos de datos referidos RRHH.
Resulta necesario para un buen análisis contar con una cantidad de registros considerables.

- Pensar **Nuevas consultas estratégicas** van a ser siempre favorables para el negocio. Patrones de consumo estacional, preferencia de productos por región, tendencias a lo largo del tiempo son ejemplos, pero además podemos sumar análisis de rentabilidad por producto y poder identificar cuáles son los más rentables y cuales podrían necesitar un ajuste de precios o costos. Se podrían segmentar los clientes, dividiendo la tabla de clientes en segmentos según su edad, sus hábitos de compras, frecuencia de visita, horario de compras, etc. Partiendo de la expansión de datos mencionada anteriormente sumando stock al sistema, se podría realizar una optimización del stock de acuerdo con la demanda de cada producto por sucursal y así poder garantizar la disponibilidad en cada sucursal.

- **Optimizar el rendimiento** pasa a ser clave en el desarrollo del sistema para asegurar una eficiencia continua a medida que la base de datos crece. Para mejorar esto, podría identificarse consultas que sean lentas y reescribirlas de manera mas eficiente.

- Relacionado al punto anterior, **Implementar índices** dentro de las tablas sería un gran aporte a la optimización del rendimiento, ya que se utilizan para encontrar rápidamente los registros que tengan un determinado valor en alguna de las columnas de la tabla. No evitamos de esta manera, que el motor de SQL Server busque un dato particular en toda la tabla (en nuestro caso es pequeña, pero se espera que crezca). En este proyecto, por ejemplo, la tabla de ordenes sería consultada con mucha frecuencia, se podrían entonces, por ejemplo, se podría implementar índices en las columnas de fechas de pedidos, y esto mejoraría el rendimiento de consultas del tipo: filtrar pedido por fecha.