**Descubriendo la Base de Datos de Fast Food**

**Nombre del autor: Minghetti Eugenia**

**Email: eugeniaminghetti@hotmail.com**

**Cohorte: DA-PT05**

**Fecha de entrega:** 14/11/2024

**Institución: FastFood**

**Texto

Descripción generada automáticamente con confianza media**

# Introducción

# FastFood es una cadena de comida rápida con diversas sucursales en el país que busca dar respuesta a ciertas preguntas sobre su funcionamiento, composición y ventas, como también comportamiento de los clientes para tomar decisiones informadas críticas para medidas futuras. Por ello, en este proyecto titulado “Descubriendo la Base de Datos de Fast Food” se buscó resolver dichas consultas.

# Se trabajó con el programa SQL Server utilizando los sublenguajes DDL y DML. El primero de ellos se empleó para crear la base de “FastFood\_DB” como también las tablas contenidas en la misma definiendo las relaciones entre dichas tablas y los campos que estaban incluidos en las mismas. Mientras que, el sublenguaje DML, se utilizó para poblar con datos las tablas y, con las tablas ya listas, permitió resolver las diversas consultas solicitadas. La complejidad de las consultas o queries fue incrementándose progresivamente al desarrollarse cada uno de los avances propuestos.

# Con las respuestas a las queries de cada avance, en conjunto con aquellas de las consultas finales, se realizaron hallazgos claves para el funcionamiento de la institución y se proporcionaron recomendaciones estratégicas. A su vez, se adicionaron detalles de cómo se optimizó la base de datos y/o las queries.

# Para finalizar se mencionan los desafíos enfrentados a lo largo de todo el proyecto y cómo se solucionaron. Por último, se cierra este informe con una reflexión personal de todo el proyecto.

# Desarrollo del proyecto

# A continuación, se desarrollarán las actividades y decisiones tomadas en cada avance.

# Avance 1º:

# Se diseñó el esquema relacional a mano de la base de datos, con sus tablas, incluyendo los campos que deberían contener y las relaciones entre las mismas, siguiendo el modelo semántico proporcionado por la institución. Posteriormente se creó la base de datos en sí (“FastFood\_DB”) y la base de datos de registro (en caso de que fuese necesario hacer algún tipo de recuperación). Una vez realizado este paso, parados sobre “FastFood\_DB” se procedió a crear cada una de las tablas con los campos correspondientes, definiendo el tipo de dato para cada columna y estableciendo las claves primarias (PK) y las claves foráneas (FK) cuando fuese necesario. Además, cuando aplicase, se utilizaron restricciones para los valores nulos en las columnas. Para el desarrollo de estas actividades se empleó el sublenguaje DDL.

# Finalizada la creación de las tablas, se chequeó que las mismas hubiesen sido creadas correctamente en la base de datos “FastFood\_DB”, y posteriormente se creó el [esquema relacional](#_Esquema_relacional_de) de la base de datos en SQLServer.

# Las queries empleadas en este avance fueron guardadas en el archivo llamado DA\_minghetti\_eugenia\_avance1PI (disponible en la carpeta de GoogleDrive). Las queries se detallarán y explicarán en la sección “[Resultados y Consultas](#_Resultados_obtenidos_de)” de este informe.

# Avance 2º:

# Usando la base de datos “FastFoof\_DB” se insertaron valores en cada una de las tablas creadas previamente y luego se chequeó que dichas tablas contengan los datos agregados. En este caso se empleó el sublenguaje DML.

# No se actualizaron ni borraron registros de las tablas ya que estas acciones modificarían los resultados obtenidos.

# Con las tablas ya pobladas de datos se procedió a responder las preguntas solicitadas en esta primera instancia por la institución a través de consultas SQL. El archivo con todas las queries, tanto aquellas empleadas para la inserción de datos en las tablas como las consultas elaboradas, fue guardado con el nombre DA\_minghetti\_eugenia\_avance2PI, y se encuentra disponible para su revisión y consulta en la carpeta de GoogleDrive. Las queries y sus respuestas se detallarán en la sección “[Resultados y Consultas](#_Resultados_obtenidos_de)” de este informe. Cabe destacar que para responder a las Preguntas 3 y 5 se plantearon dos posibles queries, la segunda de ellas en ambos casos mostrando los resultados de forma más agradable.

# Avance 3º:

# Se continuó trabajando con el sublenguale DML para responder las preguntas de la institución aplicando funciones de agregación (SUM, AVG, COUNT, MIN, MAX), incluyendo también el uso de la cláusula TOP. En este caso, las consultas a la base de datos eran un poco más complejas y buscaban obtener información más puntual sobre el negocio y su funcionamiento.

# Al igual que para el avance 2º el archivo con todas las queries fue guardado con el nombre DA\_minghetti\_eugenia\_avance3PI, y se encuentra disponible para su revisión y consulta en la carpeta de GoogleDrive. Las queries y sus respuestas, como también su explicación, se detallarán en la sección “[Resultados y Consultas](#_Resultados_obtenidos_de)” de este mismo informe.

# En este punto, si bien para responder a la consulta 5 más apropiadamente convendría agregar más datos a aciertas tablas, esta operación no se realizó para no alterar la base de datos, lo que afectaría los resultados de las siguientes queries. Además, para las preguntas 2, 3, 6 y 9 se plantearon dos posibles queries intentando lograr respuestas más agradables y completas.

# Avance 4º:

# Se utilizó el sublenguale DML para responder las preguntas planteadas por la institución combinando diferentes tablas mediante JOINS (INNER, LFET, RIGHT, FULL). Las consultas a la base de datos se complejizaban a medida que se avanzaba a lo largo de las mismas.

# Al igual que con los avances previos el archivo con todas las queries fue guardado en la carpeta de GoogleDrive con el nombre de DA\_minghetti\_eugenia\_avance4PI, y se encuentra disponible para su revisión y consulta. Las queries, sus explicaciones y respuestas se detallarán en la sección “[Resultados y Consultas](#_Resultados_obtenidos_de)” de este mismo informe.

# Debido al pequeño tamaño de nuestra base de datos, en la mayoría de las preguntas no se aprecia la diferencia entre los distintos tipos de JOINS, lo cual se modificaría si la cantidad de datos aumentase. Se proveen queries adicionales en la sección “[Optimización y Sostenibilidad](#_Optimización_y_sostenibilidad)” para algunas consultas que podrían ser de utilidad para conocer en profundidad los datos contenidos en la base de datos.

# Resultados y consultas

# Esquema relacional de la base de datos:

# Diagrama Descripción generada automáticamente

# Resultados obtenidos de las queries de los avances 1-4:

# Como primer paso, se creó la base de datos mediante la cláusula CREATE que nos permite crear nuevos objetos indicando que es una base de datos y el nombre de la misma al igual que ciertas especificaciones (el nombre lógico del archivo (NAME), la ruta completa del mismo (FILENAME), el tamaño (SIZE), el tamaño máximo que puede alcanzar (MAXSIZE) y cuánto puede aumentar el archivo en tamaño automáticamente (FILEGROWTH)). También se creó la base de datos de registro a continuación mediante la cláusula LOG ON y con las mismas especificaciones que la base de datos.

# Los comentarios al código (en verde) se han realizado con -- si es una sola línea o /\* \*/ si es más de una línea (esto aplica para todo el proyecto).

# Interfaz de usuario gráfica, Texto, Aplicación Descripción generada automáticamente

# Nos aseguramos de estar parados en la base de datos recién creada mediante la cláusula USE seguido del nombre de nuestra base de datos (al inicio de cada sesión se usa esta comando).



A continuación, creamos las tablas contenidas en FastFoodDB:

Texto

Descripción generada automáticamente

usamos la cláusula CREATE TABLE junto con el nombre que le daremos y dentro del paréntesis indicamos los campos que contendrá CategoriaID y Nombre, junto con el tipo de dato correspondiente a cada uno: INT (entero) y VARCHAR (cadena de caracteres de longitud variable), y las restricciones PRIMARY KEY (PK, clave primaria que la vinculará mediante una clave foránea a otra tabla), IDENTITY (para que se autocomplete el campo ID empezando en 1 y adicionando de a 1) y NOT NULL (restricción indicando que no puede ser nulo ese campo).

Interfaz de usuario gráfica, Texto

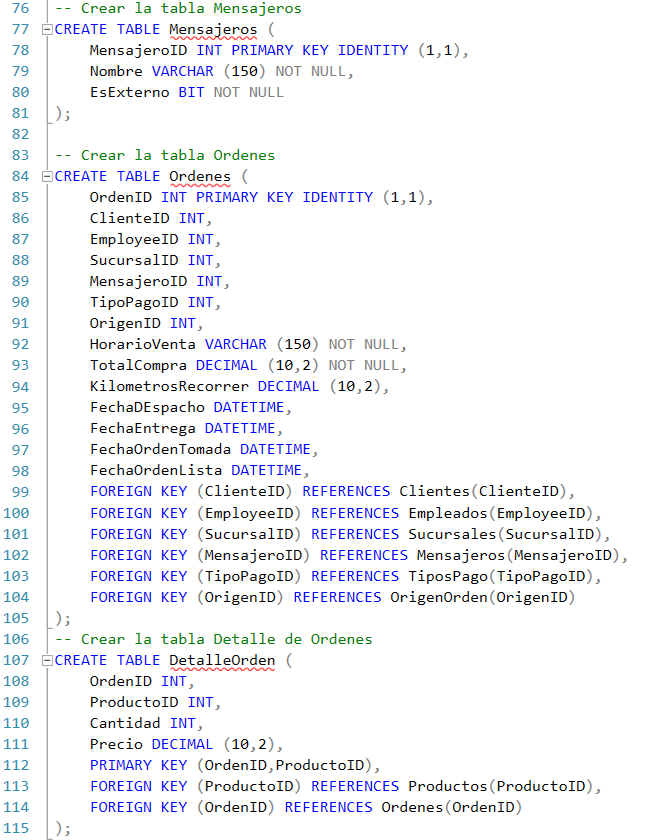
Descripción generada automáticamente

Para esta tabla procedimos de igual modo que con la anterior, solo que aquí encontramos que el campo Precio es un tipo de dato DECIMAL (de base 10 con hasta dos dígitos decimales) y tenemos la FOREIGN KEY (FK, indicamos qué columna es y con la cláusula REFRENCES con qué tabla y PK está vinculada).

Texto

Descripción generada automáticamente

Para estas tablas se procedió igual que con las anteriores.



Para la tabla Mensajeros tenemos un nuevo tipo de dato en el campo EsExterno denominado BIT (almacena 0, 1 o NULL y es útil para valores booleanos), mientras que en la tabla Ordenes encontramos DATETIME (combina la fecha y la hora). Finalmente, la tabla DetalleOrden presente una PK dada por la combinación de los registros de dos columnas OrdenID y ProductoID, los cuales combinados deben ser únicos (esta es otra manera de generar PKs sin usar una columna de identidad).

Con las tablas creadas procedemos a poblarlas con datos mediante el sublenguaje DML utilizando la cláusula INSERT INTO seguido del nombre la tabla, entre paréntesis el campo al cual agregaremos registros y VALUES, debajo colocamos los registros a insertar separados por comas y entre comillas simples si son cadenas de texto:

Texto

Descripción generada automáticamente

Es una buena práctica que una vez ejecutado el INSERT hagamos un SELECT \* FROM Nombre de la Tabla, lo cual nos permite consultar todos los registros de dicha tabla, así nos aseguramos que la inserción se realizó perfectamente. Esto se verá al final de cada sección con datos a ingresar en cada tabla.

Texto, Carta

Descripción generada automáticamente

Imagen que contiene Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Con las tablas ya pobladas de datos podemos comenzar a realizar consultas con este mismo sublenguaje (DML) para responder las preguntas de la institución:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En esta query solicitamos que se cuenten (SELECT COUNT) los registros del campo OrdenID (que al ser una PK debe ser único), al cual se le asigna el alias Ordenes mediante AS, de la tabla Ordenes indicado por el FROM. De este modo inferiormente vemos que el resultado es 10, es decir, hay 10 órdenes únicas en la tabla Ordenes.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Seleccionamos el campo Departamento y contamos (COUNT) los registros de EmployeeID (al ser una PK debe ser único) mostrado en el resultado con el alias (AS) Empleados, de la tabla (FROM) Empleados, y agrupamos (GROUP BY) por el Departamento y ordenamos (ORDER BY) por la cantidad de empleados que nos devuelve el conteo. Obtenemos así que la organización cuenta con 1 empleado en los departamentos de Cafetería, Cocina, Logística, Mantenimiento, Restaurante y Servicio, y 2 empleados en Ventas y Administración.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Esta query está organizada igual que la anterior consultando el campo CategoriaID y contando (COUNT) los registros de ProductoID (al ser una PK debe ser único) mostrado en los resultados con el alias (AS) Productos, de la tabla (FROM) Productos y agrupamos (GROUP BY) por el CategoriaID. Sin embargo, esta consulta puede optimizarse de la siguiente manera:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

de modo que se muestre el nombre de la categoría en vez del ID, para ellos debemos realizar un join implícito, es decir, realizamos el conteo (COUNT) y consultamos el nombre de la Categoria (C.Nombre, indicamos que debe usarse la columna Nombre de la tabla C), de las tablas Productos llamada P (AS P) y Categorías llamada C (AS C) (con esta forma evitamos ambigüedad si ambas tablas poseen columnas llamadas igual), luego indicamos el vínculo entre ambas tablas con la cláusula WHERE: CategoriaID de la tabla Productos es igual a CategoriaID de la tabla Categorías (P.CategoriaID=C.CategoriaID), finalmente agrupamos (GROUP BY) por el nombre de la categoría. Así obtenemos que se cuenta con 2 productos de las categorías Comida Rápida, Ensaladas, Helados, Pizzas y Postres.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

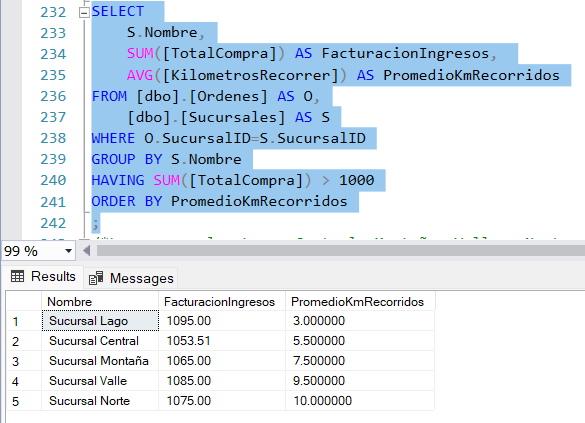
Esta query es igual que la primera, es decir contamos los ClienteID y lo llamamos (AS) Clientes en los resultados, de la tabla (FROM) Clientes. Resultando en que la cantidad de clientes es de 10.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En este caso consultamos el SucursalID, realizamos la suma (SUM) de la columna TotalCompra y la llamamos (AS) en los resultados FacturacionIngresos, y el promedio (AVG) del campo KilometrosRecorrer llamado (AS) en los resultados PromedioKmRecorridos, de la tabla (FROM) Ordenes. Agrupamos (GROUP BY) por SucursalID. Finalmente, añadimos el filtro mediante la cláusula HAVING (a lo que ya fue agrupado previamente) para que solo se muestren aquellas sucursales con una Facturacion mayor a 1000 (SUM(TotalCompra) > 1000, debemos escribirlo de este modo ya que en esta cláusula no se aceptan alias) y ordenamos por los Km recorridos ya que debíamos considerar las que los minimizaban los costos en base a ellos.

Al igual que con la query 3, esta fue optimizada del mismo modo para que se muestren los nombres de las Sucursales en vez de sus ID lo que facilita la lectura e interpretación:



De este modo podemos ver que las sucursales Lago, Central, Montaña, Valle y Norte son las que poseen un promedio de Facturación/Ingresos > $1000 y que minimizan sus costos en base a los Km recorridos, demostrando que son aquellas que presentan una mayor cantidad de ventas optimizando los recursos que disponen. Es importante destacar en este punto que sería de mucha utilidad conocer las estrategias empeladas por estas sucursales con el objetivo de ponerlas a disposición de las demás para mejorar su rendimiento en estos puntos clave.

Avanzamos con las queries al siguiente punto: funciones de agregación:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

Aquí seleccionamos la suma (SUM) de la columna TotalCompra llamada (AS) en los resultados TotalVentas de la Tabla (FROM) Ordenes, obteniendo que el Total de Ventas de $10023.51.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Seleccionamos la CategoriaID y el promedio (AVG) del Precio llamado (AS) en los resultados PromedioPrecioProducto, pero además casteamos dicho valor (CAST AS DECIMAL (10,2)) para convertirlo a un decimal de base 10 con 2 dígitos decimales, de la tabla (FROM) Productos. Agrupamos (GROUP BY) por CategoriaID y ordenamos por (ORDER BY) PromedioPrecioProducto de forma ascendente (por defecto al no hacer la aclaración).

Sin embargo, esta consulta como en otras previamente, puede optimizarse para mostrar el nombre de la Categoría y no su ID mediante un join implícito:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Así vemos que el precio promedio por categoría es $2.99 para Helado, $3.49 para Postres, $ 6.49 para Ensaladas, $8.99 para Comida Rápida y $12.49 para Pizzas.

Tabla

Descripción generada automáticamente

Es esta query calculamos el valor mínimo (MIN) llamado (AS) en los resultados ValorOrdenMinima y máximo (MAX) llamado (AS) en los resultados ValorOrdenMaxima, de las órdenes por Sucursal, por eso seleccionamos también el SucursalID, todos estos campos de la tabla (FROM) Ordenes y fueron agrupados (GROUP BY) por el SucursalID.

Es importante aclarar que nuevamente podemos optimizar la query para que se muestre el nombre de la Sucursal y no su ID, de la siguiente manera:

Imagen que contiene Tabla

Descripción generada automáticamente

De esta forma resulta que los valores mínimos y máximos son: Sucursal Bosque $900.00, Sucursal Central $1053.51, Sucursal Este $920.00, Sucursal Lago $1095.00, Sucursal Montaña $1065.00, Sucursal Norte $1075.00, Sucursal Oeste $930.00, Sucursal Playa $945.00, Sucursal Sur $955.00 y Sucursal Valle $1085.00. Como se puede notar los valores mínimos y máximos para cada sucursal son los mismos, lo cual resulta de la existencia de una única orden para cada sucursal hasta el momento.

Texto

Descripción generada automáticamente

Aquí vemos una primera forma de resolver esta query, consultando el máximo (MAX) de los KilómetrosRecorrer llamado (AS) en los resultados MayorKmRecorridos de la tabla (FROM) Ordenes para realizar la entrega de una orden.

Interfaz de usuario gráfica, Texto, Aplicación

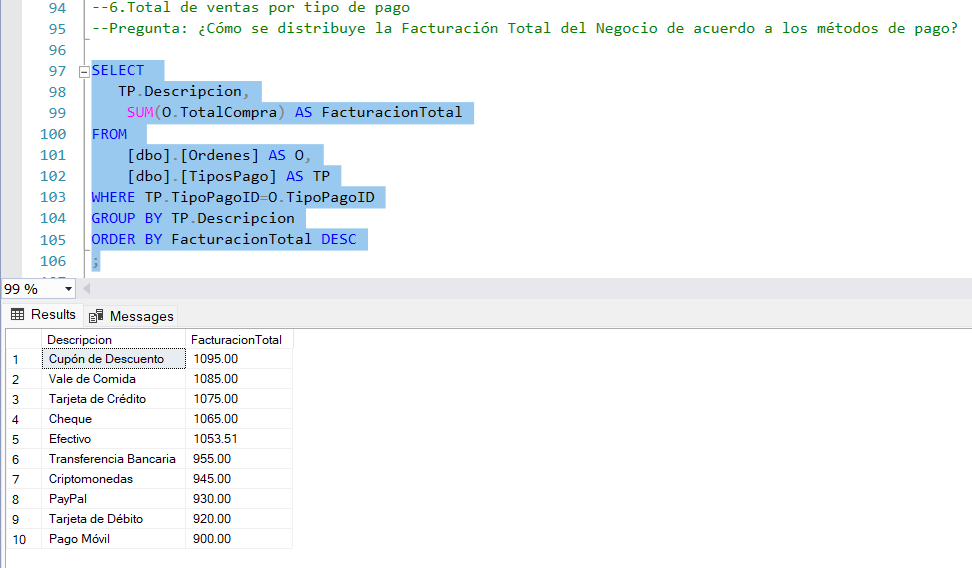
Descripción generada automáticamente

Mientras que aquí vemos otra manera de resolver la query, empleando la cláusula TOP para obtener la orden con el mayor número de Km recorridos para su entrega brindándonos más información: el OrdenID. Aquí seleccionamos el TOP 1 (un único registro), el OrdenID y los KilometrosRecorrer de la tabla (FROM) Ordenes. Para que la cláusula TOP funcione correctamente y nos dé el resultado que queremos en este caso, es indispensable ordenar los mismos (ORDER BY) por los KilometrosRecorrer de forma descendente (DESC). Como resultado se nos devuelve que la orden con OrdenID: 10 es la que presenta la mayor cantidad de km recorridos (15 km).

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para resolver esta query evitando ambigüedades y dar una mejor respuesta sería apropiados insertar más datos referentes a las otras órdenes, sin embargo, esto no se efectuó para no alterar la base de datos y los resultados de las siguientes queries. Como solo contamos con el detalle para una única orden (OrdenID: 1), el promedio de productos por orden no es un valor informativo, si contásemos con más registros para las demás órdenes seríamos capaces de obtener resultados más confiables y útiles sobre el funcionamiento de la organización. Cosultamos el OrdenID y calculamos el promedio (AVG) de la Cantidad llamada (AS) en los resultados CantidadPromedioProductos de la tabla (FROM) DetalleOrden y agrupamos (GROUP BY) por el OrdenID,



En esta query, si consideramos que la distribución de la Facturación Total puede obtenerse mediante una suma, consultamos la Descripción del tipo de pago (TP.Descripción) de la tabla (FROM) Tipos de Pago (llamada TP: AS TP) y realizamos la suma (SUM) del TotalCompra (O. TotalCompra) llamado (AS) en los resultados FacturacionTotal de la tabla (FROM) Ordenes (llamada O: AS O), para así obtener el nombre del Tipo de Pago y no su ID mediante un join implícito (WHERE TP.TipoPagoID=TipoPagoID). Agrupamos (GROUP BY) por TP.Descripción y ordenamos (ORDER BY) por la FacturacionTotal de forma descendente (DESC), obtenemos que: la Facturación Total de la organización según los métodos de pago se distribuye de la siguiente manera Cupón de Descuento, Vale de comida, Tarjeta de Crédito, Cheque, Efectivo, Transferencia Bancaria, Criptomonedas, Paypal, Tarjeta de Débito y Pago Móvil.

Sin embargo, si consideramos que la distribución de la Facturación Total se expresa mejor mediante promedio (AVG) o frecuencia de los registros (COUNT), la query puede plantearse de la siguiente manera:

Tabla

Descripción generada automáticamente con confianza media

Solo se agregó a la consulta el cálculo del promedio casteado (como se explicó para una query anterior) y el conteo. Ya que la consulta no aclara que opción se prefiere para ejemplificar la distribución aquí quedan varias posibilidades.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

En esta query buscamos obtener la Sucursal con el ingreso promedio más alto, para lo que usamos la cláusula TOP 1 (queremos una única sucursal) y calculamos el promedio (AVG) casteado (igual que previamente) llamado (AS) en los resultados IngresoPromedio. Igual que anteriormente con TOP debemos ordenar (ORDER BY) para que los resultados muestren adecuadamente lo que deseamos, en este caso por el IngresoPromedio de manera descendente (DESC). Así, visualizamos que la sucursal 9 presenta el Ingreso promedio más alto con $1095.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En este caso seleccionamos el Nombre de la sucursal (S.Nombre) llamado (AS) en los resultados NombreSucursal, realizamos la suma (SUM) de la columna TotalCompra y la llamamos (AS) en los resultados VentasTotales de la tabla (FROM) Ordenes (llamada O: AS O) y de la tabla Sucursales (llamada S: AS S), para vincular ambas tablas mediante un join implícito y poder visualizar el nombre de la sucursal y no su ID (WHERE O.SucursalID=S.SucursalID). Agrupamos (GROUP BY) por S.Nombre, añadimos el filtro mediante la cláusula HAVING (a lo que ya fue agrupado previamente) para que solo se muestren aquellas sucursales con Ventas Totales mayores a 1000 (SUM (TotalCOmpra) >1000) y ordenamos (ORDER BY) por las VentasTotales.

De este modo podemos ver que las sucursales Central Montaña, Norte, Valle y Lago son las que poseen ventas totales mayores a $1000.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

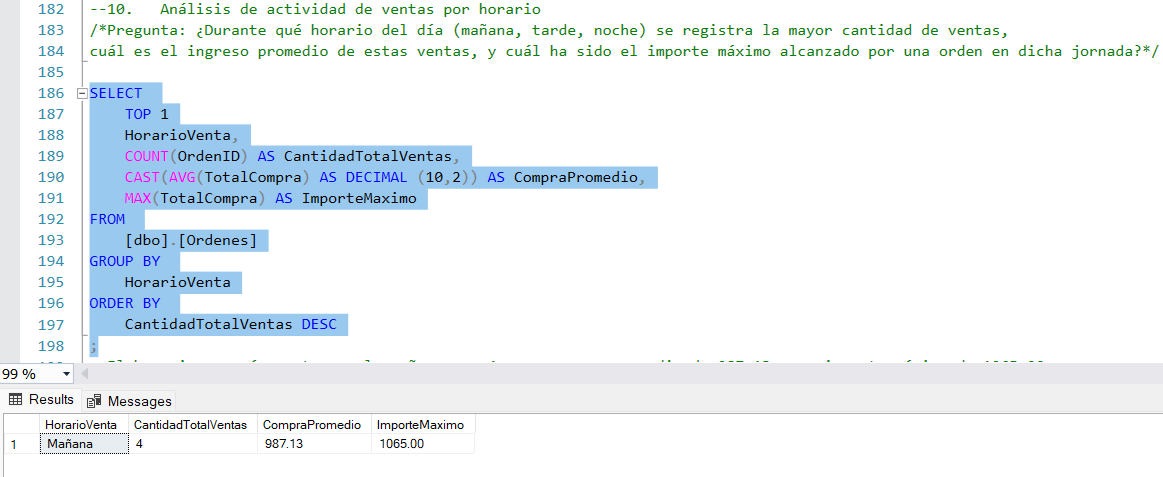
Descripción generada automáticamente

Para resolver esta query podemos hacerlo mediante dos queries independientes, ejecutándolas al mismo tiempo y mostrando dos tablas. En la primer query se calcula el promedio (AVG) de TotalCompra, casteado a DECIMAL, antes del 1 de julio usando como filtro dicha fecha (WHERE FechaOrdenTomada < ‵2023-07-01′); mientras que en la segunda query se calcula el promedio (AVG) de TotalCompra, casteado a DECIMAL, después del 1 de julio usando como filtro dicha fecha (WHERE FechaOrdenTomada >= ‵2023-07-01′). La venta promedio antes del 1 de julio es de $979.75, mientras que después del 1 de julio es de $1036.25.

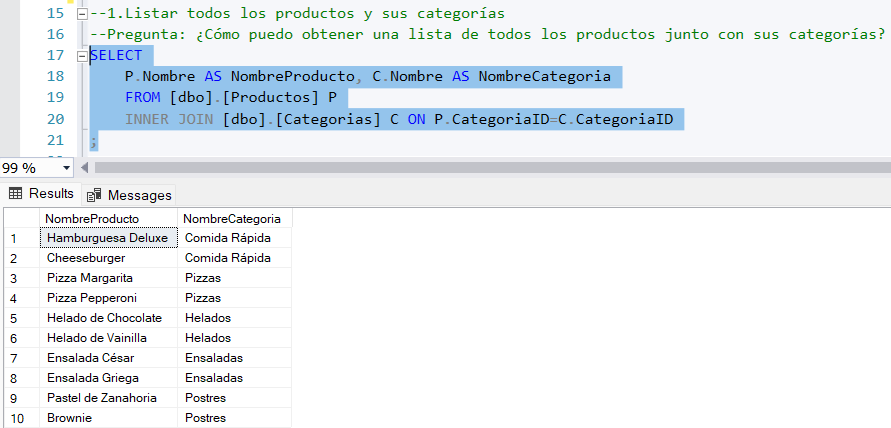
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En esta variante, la query incluye una condición mediante la cláusula CASE, en la cual mediante WHEN indicamos la primer condición FechaOrdenTomada < ‵2023-07-01′ y que en ese caso (THEN) ponga en la tabla resultante Antes del 1 de julio de 2023, sino (ELSE) debe indicar Después del 1 de julio de 2023, finalizamos con END AS para que esa columna creada se llame periodo; posterior a esta cláusula calculamos el promedio (AVG) de TotalCompra casteado y agrupamos (GROUP BY) por la condición del CASE (debemos volver a escribirla completa porque no se aceptan alias en esta cláusula). Usando esta segunda opción obtenemos una única tabla con los resultados.



En esta query utilizamos la cláusula TOP 1 (ya que nos interesa el horario con mayor cantidad de ventas), seleccionamos el HorarioVenta, contamos (COUNT) el OrdenID y lo llamamos (AS) en los resultados CantidadTotalVentas, calculamos el (AVG) de TotalCompra casteado y lo llamamos (AS) en los resultados CompraPromedio y obtenemos el máximo (MAX) de TotalCompra y lo llamamos (AS) en los resultados ImporteMaximo, todo de la tabla (FROM) Ordenes. Agrupamos (GROUP BY) por Horario Venta y ordenamos (ORDER BY) (para que TOP nos muestre el resultado que buscamos) por CantidadTotalVentas de forma descendente (DESC). Como resultado obtenemos que el horario con más ventas (un total de 4) es la mañana, siendo la compra promedio de $987.13 y el importe máximo de $1065.00.

Avanzamos con las queries aplicando distintos tipos de JOINS para poder mostrar la información contenía en diferentes tablas: 

En esta query necesitamos mostrar la información de dos tablas Productos y Categorías, para ello hacemos un INNER JOIN ya que nos interesa la información compartida por ambas tablas: productos y sus categorías. Consultamos el nombre del producto (P.Nombre) llamado (AS) en los resultados NombreProducto y el nombre de la categoría (C. Nombre) llamado (AS) en los resultados NombreCategoria, de la tabla (FROM) Productos llamada P y ligada a la tabla categorías llamada C mediante un INNER JOIN en (ON) P.CategoriaID=C.CategoriaID (campo coincidente entre ambas tablas y que permite la vinculación). Como resultado vemos la categoría a la que pertenece cada producto.

# Interfaz de usuario gráfica, Aplicación Descripción generada automáticamente

# En este caso, también podemos realizar un INNER JOIN entre las tablas Empleados (alias: E) y Sucursales (alias: S) ya que queremos ver a qué sucursal está asignado cada empleado. Al igual que en la query anterior queremos ver el nombre del empleado (E.Nombre) llamado (AS) en los resultados NombreEmpleado, su posición (E.Posicion), el departamento al que pertenece (E.Departamento) y el nombre de la sucursal (S.Nombre) llamado (AS) en los resultados NombreSucursal. Hacemos un INNER JOIN de la tabla E con S en (ON) E.SucursalID= S.SucursalD (campo coincidente entre ambas tablas y que permite la vinculación). Como resultado vemos que todos los empleados pertenecen a la Sucursal Central.

# Si observamos no hay ningún empleado sin sucursal asignada, pero si tenemos muchas sucursales sin empleados asignados (lo cual podría ser un error que debe corregirse), para ver esto podemos realizar un RIGHT JOIN:

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

La query es exactamente la misma solo que hacemos un RIGHT JOIN en vez de un INNER JOIN, el cual nos trae todas las columnas indicadas de la tabla derecha (en este caso sucursales) y completa con NULL cuando no hay registros en la tabla de la izquierda (Empleados) para ese campo.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

En esta query nos interesa conocer si todos los productos tienen una categoría asignada por lo que consultamos el nombre del producto (P.Nombre) llamado (AS) en los resultados NombreProducto y el nombre de la categoría (C.Nombre) llamado (AS) en los resultados NombreCategoria, vinculando la tabla Productos (alias P) y la tabla Categorías (alias C) mediante un LEFT JOIN a través de (ON) P.CategoriaID=C.CategoriaID (campo coincidente entre ambas tablas y que permite la vinculación). En este caso como queremos obtener todos los registros de la tabla izquierda (Productos) y las coincidencias con la tabla de la derecha (Categorias) empleamos este tipo de join. Como podemos observar todos los productos están asignados a una categoría.

Sin embargo, para saber si hay categorías sin productos asignados podemos realizar un RIGHT JOIN:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

La consulta es exactamente igual, solo cambia el tipo de join, y resulta en que hay categorías sin productos asignados, que sería de gran utilidad cargar en la base de datos para completar la información.

Aplicación

Descripción generada automáticamente con confianza baja

En este caso consultamos el OrdenID (O.OrdenID), el Total de la compra (O.TotalCompra), la fecha de la orden (O.FechaOrdenTomada), el horario de la venta (O.HorarioVenta), el nombre del cliente (C.Nombre) llamado (AS) en los resultados NombreCliente, el nombre y la posición del empleado (E.Nombre y E.Posicion respectivamente) llamados (AS) en los resultados NombreEmpleado y PosicionEmpleado, y el nombre del mensajero (M.Nombre) llamado (AS) en los resultados NombreMensajero de la tabla Ordenes (alias O), vinculada a las tablas Clientes (alias C), Empleados (alias E) y Mensajeros (alias M) mediante INNER JOIN en todos los casos, a través de (ON) los siguientes campos compartidos entre la tabla Ordenes: C.ClienteID=O.ClienteID (con Clientes), E.EmployeeID=O.EmployeeID (con Empleados) y M.MensajeroID=O.MensajeroID (con Mensajeros). Como resultado vemos una tabla con todos los datos de interés.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para responder esta pregunta debemos elaborar una query consultando el nombre de la categoría y de la sucursal (C.Nombre y S.Nombre respectivamente) llamados (AS) en los resultados NombreCategoria y NombreSucursal, y también la suma (SUM) de la Cantidad de productos vendidos llamado (AS) en los resultados TotalProductosVendidos, vinculamos la tabla DetalleOrden (alias D) mediante INNER JOIN con las tablas Ordenes (alias O) por (ON) D.OrdenID=O.OrdenID, Productos (alias P) por (ON) P.ProductoID=D.ProductoID, Categorías (alias C) por (ON) C.CategoriaID=P.CatgeoriaID, y Sucursales por (ON) S.SucursalID=O.SucursalID. Luego agrupamos por S.Nombre y C.Nombre, y ordenamos por NombreCategoria y NombreSucursal. Así vemos que en la Sucursal Central las categorías de productos más vendidas son Comida Rápida y Pizzas, tal vez sería posible plantear alguna promoción o descuento para las demás categorías buscando aumentar sus ventas en esta sucursal.

# Consultas finales:

* + **Eficiencia de los mensajeros:** [**¿Cuál es el tiempo promedio desde el despacho hasta la entrega de los pedidos gestionados por todo el equipo de mensajería?**](#_Eficiencia_de_los)
  + **Análisis de Ventas por Origen de Orden:** [**¿Qué canal de ventas genera más ingresos?**](#_Análisis_de_Ventas)
  + **Productividad de los Empleados:** [**¿Cuál es el nivel de ingreso generado por Empleado?**](#_Productividad_de_los)
  + **Análisis de Demanda por Horario y Día:** [**¿Cómo varía la demanda de productos a lo largo del día?**](#_Análisis_de_Demanda) **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.**
  + **Comparación de Ventas Mensuales:** [**¿Cuál es la tendencia de los ingresos generados en cada periodo mensual?**](#_Comparación_de_Ventas)
  + **Análisis de Fidelidad del Cliente:** [**¿Qué porcentaje de clientes son recurrentes versus nuevos clientes cada mes?**](#_Análisis_de_Fidelidad) **NOTA: La consulta se enfocaría en la frecuencia de órdenes por cliente para inferir la fidelidad.**

# Hallazgos clave

# Eficiencia de los mensajeros:

Texto

Descripción generada automáticamente con confianza baja

Para responder esta consulta, se elaboró una query que calcula el promedio AVG de la diferencia (DATEDIFF) en minutos (extracción de los minutos de la FechaDEspacho y FechaEntrega con MINUTE) denominado (AS) en los resultados TiempoPromedio de la tabla (FROM) Ordenes.

Todo el equipo de mensajería entrega la orden en un tiempo promedio de 30 minutos desde su despacho.

# Análisis de Ventas por Origen de Orden:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En esta consulta usamos la cláusula TOP 1 (porque nos interesa el canal con más ventas), la descripción (OO.Descripción) llamada (AS) en los resultados CanalVenta y la suma (SUM) del TotalCompra llamada (AS) en los resultados IngresoTotal, de la tabla (FROM) Ordenes (alias O: AS O) vinculada mediante un JOIN con la tabla OrigenOrden (alias OO: AS OO) mediante (ON) O.OrigenID=OO.OrigenID (campo compartidos entre las tablas que permite la vinculación). Agrupamos (GROUP BY) por OO.Descripción y ordenamos (ORDER BY) por IngresoTotal de forma descendente (DESC).

Obtenemos que el Canal de ventas con más ingresos es Presencial por un total de $2140.

# Productividad de los Empleados:

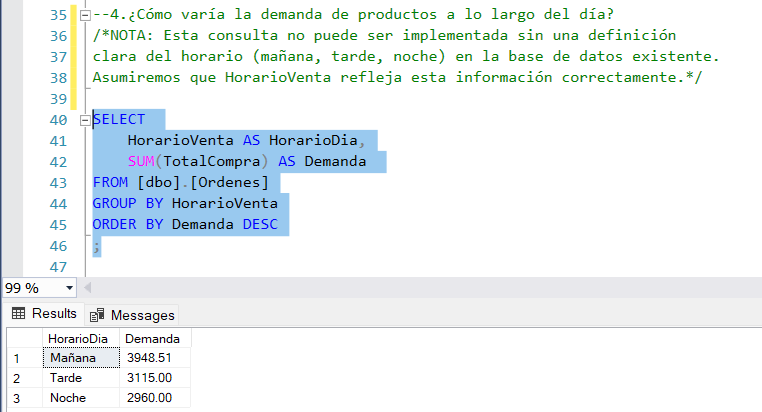
Imagen que contiene Texto

Descripción generada automáticamente

Para resolver esta query consultamos el nombre del empleado y su posición (E.Nombre y E.Posicion) denominados (AS) en los resultados NombreEmpleado y POsicionEmpleado respectivamente, y sumamos (SUM) el TotalCompra denominado (AS) en los resultados IngresoTotal, de la tabla (FROM) Ordenes (alias O: AS O) vinculada (JOIN) a la tabla Empleados (alias E: AS E) mediante (ON) O.EmployeeID=E.EmployeeID (campo compartidos entre las tablas que permite la vinculación). Agrupamos (GROUP BY) por E.Nombre y E.Posicion y ordenamos (ORDER BY) por IngresoTotal de forma descendente (DESC).

Así vemos los ingresos generados por cada empleado y sus posiciones. Podemos comprender cómo se distribuyen los ingresos generados entre las diferentes posiciones.

# Análisis de Demanda por Horario y Día:



Asumiendo que HorarioVenta refleja correctamente el horario lo consultamos a través de la query y lo denominamos (AS) en los resultados HorarioDia y sumamos (SUM) TotalCompra denominado (AS) en los resultados Demanda, de la tabla (FROM) Ordenes, agrupamos (GROUP BY) por HorarioVenta y ordenamos (ORDER BY) por Demanda de forma descendente (DESC).

Así vemos que la mañana es el horario con una mayor demanda, seguida por la tarde y último la noche.

# Comparación de Ventas Mensuales:

Tabla

Descripción generada automáticamente con confianza baja

En este caso extraemos el mes (MONTH) de la FechaOrdenTomada y lo denominamos (AS) en los resultados Mes, sumamos (SUM) el TotalCompra denominado (AS) en los resultados IngresoMensual y el promedio (AVG) casteado a decimal en base 10 y con dos dígitos decimales, y lo denominamos (AS) en los resultados IngresoPromedioMensual, de la tabla (FROM) Ordenes. Agrupamos (GROUP BY) por MONTH(FechaOrdenTomada) y ordenamos (ORDER BY) por IngresoMensual de forma ascendente (no es necesario indicarlo).

Como podemos ver los meses con ingresos menores a $1000 son Octubre, Marzo, Abril, Junio y Mayo. Aumentar las ventas en estos meses sería una gran oportunidad, tal vez celebrando algún día especial como el Dia de la Madre (en Argentina) con la aplicación de diferentes promociones.

# Análisis de Fidelidad del Cliente:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

En esta primera parte de la query se implementó un CTE (Common Table Expressions) para definir un conjunto de datos temporales que permite organizar mejor la consulta y simplificarla, funcionan como cualquier otra tabla de la base de datos.

Creamos el CTE PrimerPedido (WITH PrimerPedido AS) dentro de los paréntesis colocamos la consulta a SQL que va a representar, en este caso, la consulta del ClienteID y el cálculo de la fecha mínima (MIN (CAST (FechaOrdenTomada AS DATE)) AS PrimerPedido) [es decir casteamos la fecha de toma de la orden de DATETIME a DATE y calculamos cuál es la primer fecha de orden de ese cliente denominada PrimerPedido], de la tabla (FROM) Ordenes y agrupamos por (GROUP BY) ClienteID.

Debe ejecutarse en conjunto con nuestra consulta que se escribe a continuación y es la siguiente:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Aquí vemos que consultamos el mes (MONTH) de O.FechaOrdenTomada denominada (AS) en los resultados Mes, y contamos:

-cuántos clientes nuevos hay (COUNT): el DISTINCT nos asegura que se cuente cada cliente una vez por mes, y usamos la cláusula CASE para que si la fecha de la orden tomada es igual a la fecha del PP.PrimerPedido entonces (THEN) el ClienteID es nuevo (AS NuevoCliente). Usamos el (CAST(O.FechaOrdenTomada AS DATE)) para convertir DATETIME a DATE igual que en el CTE.



-cuántos clientes recurrentes hay (COUNT): el DISTINCT nos asegura que se cuente cada cliente una vez por mes, y usamos la cláusula CASE para que si la fecha de la orden tomada es mayor a la fecha del PP.PrimerPedido entonces (THEN) el ClienteID es recurrente (AS ClienteRecurrente). Usamos el (CAST(O.FechaOrdenTomada AS DATE)) para convertir DATETIME a DATE.



- cuántos clientes hay (COUNT) en total: el DISTINCT nos asegura que se cuente cada cliente una vez y los denominamos (AS) en los resultados TotalClientes.



Continuamos calculando el porcentaje de clientes nuevos y recurrentes ((clientes nuevos o recurrentes dividido el total de clientes)\*100), pero para ello es necesario castear el conteo previo como FLOAT porque sino se producen resultados erróneos (al dividir enteros en enteros nos resulta enteros, lo que puede llevar a equivocaciones en los resultados del porcentaje), y lo denominamos (AS) en los resultados PorcentajeNuevosClientes y PorcentajeClientesRecurrentes:



Finalmente colocamos de qué tabla proviene: (FROM) Ordenes (alias O: AS O), vinculada (JOIN) con el CTE PrimerPedido llamado PP (AS PP) mediante O.ClienteID=PP.CLienteID, agrupamos (GROUP BY) por (MONTH)O.FechaOrdenTomada y ordenamos (ORDER BY) por Mes.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Como resultado podemos observar que hasta el momento siempre tenemos clientes nuevos cada mes, por lo que beneficiar a nuestros clientes previos sería un gran incentivo para que consuman nuevamente en las sucursales.

# Recomendaciones estratégicas

-Conocer las estrategias empeladas por las sucursales Lago, Central, Montaña, Valle y Norte que son las que presentan una mayor cantidad de ventas optimizando los recursos que disponen (minimización de km recorridos por entrega), con el objetivo de implementarlas en las demás sucursales, cuando fuese posible, buscando mejorar el rendimiento de las mismas en estos puntos clave.

- Teniendo en cuenta que la Facturación Total de la organización según los métodos de pago se distribuye de la siguiente manera (solo mencionando los tres primeros métodos con más facturación): Cupón de Descuento, Vale de comida y Tarjeta de Crédito, sería de utilidad diseñarse alguna promoción vinculada a estos medios de pago buscando aumentar aún más su uso. Además, ofrecer algún beneficio para otros métodos de pago sería de gran interés si la institución desea incrementar el uso de los mismos.

-Considerando que la Sucursal Lago (SucursalID: 9) es aquella con el ingreso promedio más alto, sería indispensable comprender los motivos de esta situación teniendo en cuenta cuáles son sus estrategias de captación de clientes, los beneficios que brindan a los mismos, las promociones en vigencia y la atención brindada a los clientes, ya que toda esta información podría ser aplicada en las demás sucursales aumentando también sus ingresos.

-Recomendaría actualizar la base de datos incorporando los registros, si están disponibles, correspondientes a los empleados de todas las sucursales exceptuando la Central (la cual ya están ingresados en la base de datos), y también los productos en las categorías sin registros aún. De este modo, la base de datos posee toda la información útil para realizar cualquier análisis que nos interese.

- Observando que en la Sucursal Central las categorías de productos más vendidas son Comida Rápida y Pizzas, considero que desarrollar y ofrecer promociones o descuentos para las demás categorías sería una buena estrategia para incrementar sus ventas en esta sucursal.

**Respecto a las consultas finales:**

-Si bien el equipo de mensajería entrega la orden en un tiempo promedio de 30 minutos desde su despacho debería indagarse qué equipos se utilizan para la entrega y si una mejora en los mismos podría reducir este tiempo promedio.

-El Canal de venta Presencial es el que genera más ingresos, por lo que el desarrollo de un plan de beneficios para los demás canales de venta podría incrementar su uso.

-Evaluar la distribución de ingresos entre las posiciones de los empleados y analizar motivos de las diferencias halladas para entender las preferencias de los clientes por ciertos empleados y/o posiciones.

-Considerando que la mañana es el horario con una mayor demanda respecto a ventas podría implementarse una modificación de los horarios de apertura y cierre del local para aprovechar esta situación. Si el objetivo es incrementar las ventas en los horarios tarde y noche, la aplicación de promociones y beneficios para estos momentos del día podría atraer más clientes y aumentar las ventas.

-La tendencia de ingresos por mes varía siendo Septiembre el mes con mayores ingresos y Octubre el que posee menos. Aprovechar fechas festivas y/o especiales en los cinco meses con menos ingresos mediante el desarrollo de promociones atractivas para los clientes podría aumentar considerablemente los ingresos en dichos meses.

-Desarrollo e implementación de un programa de beneficios para clientes previos con el objetivo de lograr que vuelvan a consumir en las sucursales, ya que hasta el momento solo se registran nuevos clientes cada mes.

# Optimización y sostenibilidad

# -Especificaciones de los tipos de datos en la creación de las tablas para reducir el espacio en memoria que ocupan nuestros datos.

# -Cada archivo de query inicia con la cláusula de USE FastFoodDB para asegurarnos estar posicionados en la base de datos correcta.

# -Optimización de consultas en aquellas preguntas en que fuese posible de modo que los resultados fuesen más amigables con los lectores de la información y se facilitara el entendimiento de los mismos, por ejemplo (ver sección “[Resultados y Consultas](#_Resultados_obtenidos_de)” para detalles de las queries): Preguntas 3 y 5 del 2º Avance obteniendo para la primera los nombres de la categorías asociados a los productos, y para la segunda los nombres de las sucursales con un promedio de Facturación/Ingresos superior a $1000.00 que minimizan sus costos en base al promedio de km recorridos; preguntas 2 (obtención del nombre de la categoría asociado al precio promedio de los productos), 3 (obtención del nombre de la sucursal y el valor de la orden mínima y máxima), 6 (contemplar todas las posibilidades de referirse a la distribución de la facturación total según el método de pago) y 9 (obtención de una única tabla con los resultados) del 3º Avance.

-Generación de queries adicionales en el caso que se desee ahondar más profundamente en los datos disponibles y también para la identificación de aquellas partes de la base de datos que deben ser actualizadas o a las que falta aún ingresar registros. Este es el caso de las queries 2 (empleados asignados a sucursales, identificación de sucursales que no tienen empleados asignados) y 3 (productos asignados a categorías, identificación de categorías sin productos asignados) del 4º Avance.

# -Creación de un CTE (Common Table Expressions) para la quey 6 de las Consultas Finales, con el objetivo de definir un conjunto de datos temporales que pueden ser llamados posteriormente como si fuese otra tabla, que permitió organizar mejor la consulta y de forma más simple y legible.

-Creación de vistas para guardar y reutilizar consultas complejas de forma sencilla facilitando el acceso a las mismas y también resguardar los datos regulando qué usuarios acceden a qué información.

-El rendimiento de la base de datos respecto a su funcionamiento para las consultas ha sido óptimo.

# Desafíos y soluciones

# Respecto al proyecto en sí, algunas de las preguntas planteadas eran ambiguas y no quedaba claro qué se debía responder:

# -Pregunta 5 del 3º Avance para obtener una respuesta acorde se plantea como solución la posibilidad de incorporar más registros relativos a otras órdenes, entendiendo de este modo el correcto funcionamiento de la query y evitando ambigüedades en su construcción e interpretación.

# -Pregunta 6 del 3º Avance para evitar que falte información respecto a la distribución de la facturación total se incorpora en la query tres formas distintas de poder obtenerla: suma, promedio y conteo de registros.

Respecto a lo personal, este proyecto me llevó a interiorizarme con mayor profundidad en algunos temas tratados para poder aplicarlos correctamente. Además, necesité practicar por mi cuenta el desarrollo de consultas SQL especialmente cuando requerían cierto grado de complejidad. Fue un tema completamente nuevo para mí, que disfruté aprender, y en el que me autodesafiaba a mejorar a cada paso. Me permitió además entender el gran potencial que tiene la IA para explicar conceptos y ayudarme a mejorar mis queries. Sin duda, fue un gran módulo sobre el que continuaré aprendiendo y practicando.

# Reflexión personal

# A lo largo de este proyecto fui capaz de entender el funcionamiento de las bases de datos SQL (desde el principio: creando la base de datos y todo su contenido hasta la realización de consultas que aumentaban en complejidad), poniendo en práctica lo visto en las lectures de forma amena y divertida, desafiándome a cada paso para lograr el objetivo. Sin duda estos conocimientos son indispensables en la carrera de cualquier Analista de Datos y sientan las bases para trabajar con diversos softwares SQL. Además, se desarrollan habilidades blandas de comunicación, vinculación y trabajo en equipo, al colaborar en conjunto con mis compañeros para resolver las dudas que surgían y los problemas que teníamos, compartiendo diversas soluciones igualmente posibles y aplicables, como también explicando algunos contenidos de las lectures.

# Si tuviera que volver a empezar el proyecto mantendría el iniciar elaborando un esquema de las tablas a incluir en la base de datos y las relaciones entre ellas, ya que de esa manera se sigue un orden lógico de creación y se evitan problemas cuando las tablas tienen PKs y FKs que las vinculan. La creación de las tablas y la inserción de datos en las mismas, podría hacerse en conjunto ya que de esta manera es más fácil visualizar qué estamos cargando en cada tabla y tenemos en mente su estructura. Respecto a las consultas, considero que ir complejizándolas al avanzar en el proyecto es la mejor forma de entenderlas y ser capaces de aplicarlas, claramente, muchas de las consultas realizadas en la primera clase pueden resolverse más sencillamente con contenido de las clases siguientes, pero considero que eso es lo interesante, ver cómo podemos mejorar nuestras consultas al incorporar conceptos nuevos.

# EXTRA CREDIT

# -Se plantearon mejoras en las siguientes queries 3 y 5 del 2º Avance; 2, 3, 6 y 9 del 3º Avance brindando mejores respuestas con la información solicitada.

# -Se plantearon queries adicionales en las queries 2 y 3 del 4º Avance para brindar más información sobre los datos presentes en la base datos, posibilitando entender que áreas aún requieren la inserción de registros.

-Se desarrolló un CTE para la query 6 de las consultas finales buscando simplificar la consulta y organizarla mejor para que su comprensión sea más sencilla.