MODELO DE BASE DE DATOS RELACIONAL

DE UNA FERRETERIA

LEONARDO JAVIER TORRES VELILLA

UNIVERSIDAD PILOTO DE COLOMBIA

ASIGNATURA METODOLOGIAS Y HERRAMIENTAS PARA EL TRATAMIENTO Y LA ANALITICA DE DATOS

BOGOTA D.C

Tabla de Contenido

1. INTRODUCCIÓN

La gestión eficiente de la información es un factor crítico para la toma de decisiones estratégicas en cualquier organización. En el caso de una ferretería con operaciones diversificadas en distintas oficinas regionales, resulta fundamental contar con una base de datos bien estructurada que permita almacenar, consultar y analizar los distintos procesos que conforman el negocio: atención a clientes, ventas, control de inventarios, desempeño de empleados y distribución geográfica de oficinas.

Este proyecto presenta el análisis de una base de datos relacional desarrollada para una ferretería, la cual integra tablas esenciales que modelan las entidades principales del negocio: **clientes**, **empleados**, **oficinas**, **pedidos** y **productos**. Cada una de estas entidades está conectada mediante relaciones bien definidas que garantizan la integridad referencial, lo que permite un flujo ordenado de información y consultas eficientes.

A través de este análisis se han diseñado múltiples consultas SQL orientadas a extraer conocimiento valioso del sistema, tales como el rendimiento de los vendedores, el cumplimiento de metas por oficina, la rentabilidad de productos, la frecuencia de pedidos y la actividad de los clientes. El proceso ha sido acompañado por un diseño lógico y conceptual del modelo de datos, validando su normalización para evitar redundancias y asegurar la consistencia.

El objetivo final es proporcionar una visión integral del funcionamiento del sistema comercial de la ferretería, identificar oportunidades de mejora y formular estrategias en base a datos que permitan potenciar el crecimiento, optimizar recursos y elevar la competitividad del negocio a corto, mediano y largo plazo.

Para la creación de esta base de datos usamos la herramienta MySQL Workbench, la cual es ideal para conectar, administrar y realizar tareas de bases de datos. En ella crearemos nuestro diagrama entidad relación (ER).

2. DEFINICIONES

Antes de entrar en el detalle de como se crearon las tablas hablaremos un poco acerca de lo que es MYSQL y sus principales características:

1) ¿Qué es MYSQL?

   Es un sistema de administración de base de datos relacional de código abierto basado en el lenguaje estructurado de consulta (SQL), almacena los datos en tablas separadas y de forma organizada en vez de colocar todos los datos en un solo lugar. *(Jeffrey Erickson, 2024, agosto 29).*

2) ¿Qué es DML?

   El lenguaje de manipulación de datos es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios de esta llevar a cabo las tareas de consulta o manipulación de los datos, organizados por el modelo de datos adecuado.
   El lenguaje de manipulación de datos más popular hoy día es SQL, usado para recuperar y manipular datos en una base de datos relacional. Otros ejemplos de DML son los usados por bases de datos IMS/DL1, CODASYL u otras.

   Las instrucciones DML incluyen:

- **INSERT**: agrega uno o más registros a una (y sólo una) tabla en una base de datos relacional.
- **UPDATE**: modificar los valores de un conjunto de registros existentes en una tabla.
- **DELETE**: borra cero o más registros existentes en una tabla.
- **SELECT**: Las consultas en SQL constan de uno o más bloques de recuperación SELECT-FROM-WHERE. SELECT(elegir) atributos; FROM(de) relaciones; WHERE(donde) condiciones lógicas.

3) que es DDL? Ejemplos

El lenguaje de definición de datos es el que se encarga de la modificación de la estructura de los objetos de la base de datos. Existen cuatro operaciones básicas: CREATE, ALTER, DROP y TRUNCATE.

- **CREATE:** Este comando crea un objeto dentro de la base de datos. Puede ser una tabla, vista, índice, trigger, función, procedimiento o cualquier otro objeto que el motor de la base de datos soporte.

  (crear una tabla)
- **ALTER:** permite modificar la estructura de un objeto.

  (agregar columna a una tabla)
- **DROP:** Este comando elimina un objeto de la base de datos ya sea un campo o una tabla.
- **TRUNCATE:** Este comando trunca todo el contenido de una tabla. La ventaja sobre el comando DELETE, es que, si se quiere borrar todo el contenido de la tabla, es mucho más rápido.

3. DICCIONARIO DE DATOS

CLIENTES			
nombre campo	tipo	observación	Descripción Campo
numclie	VARCHAR(40)	PK	Numero de cliente (único)
nombre	VARCHAR(50)		Nombre del cliente
repclie	CHAR(4)		Representante que atiende al cliente
limitecredito	INT(10)		limite crédito (valor)

EMPLEADOS			
nombre campo	tipo	observación	Descripción Campo
numemp	CHAR(3)	PK	Numero de empleado (único)
nombre	VARCHAR(40)		Nombre del empleado
edad	INT(3)		edad del empleado
oficina	CHAR(2)		numero de la oficina
titulo	VARCHAR(20)		Representante
contrato	DATE		fecha de contrato
jefe	CHAR(3)		numero de jefe
cuota	INT(10)		valor
ventas	INT(10)		valor

OFICINAS			
nombre campo	tipo	observación	descripción Campo
oficina	CHAR(2)	PK	numero de la oficina (Único)
ciudad	VARCHAR(20)		ciudad oficina
región	VARCHAR(10)		región en la que se encuentra
dir	CHAR(3)		numero de dirección
objetivo	INT(10)		objetivo por conseguir
ventas	INT(10)		valor de venta

PEDIDOS			
nombre campo	tipo	observación	descripción Campo
codigo	INT(10)	PK	código único
numpedido	INT(10)		numero de pedido
fechapedido	DATE		fecha de pedido
clie	VARCHAR(40)		cliente que hace pedido
rep	CHAR(3)		Representante que vende
fab	CHAR(3)		Código fabricante
producto	CHAR(10)		descripción producto
cant	INT(10)		cantidad de pedido
importe	INT(10)		valor

PRODUCTOS			
nombre campo	tipo	observación	descripción Campo
idfab	CHAR(3)	PK	Código único fabricante
idproducto	VARCHAR(10)	PK	Código único producto
descripcion	VARCHAR(30)		descripción del producto
precio	INT(10)		valor
existencias	INT(10)		cantidad

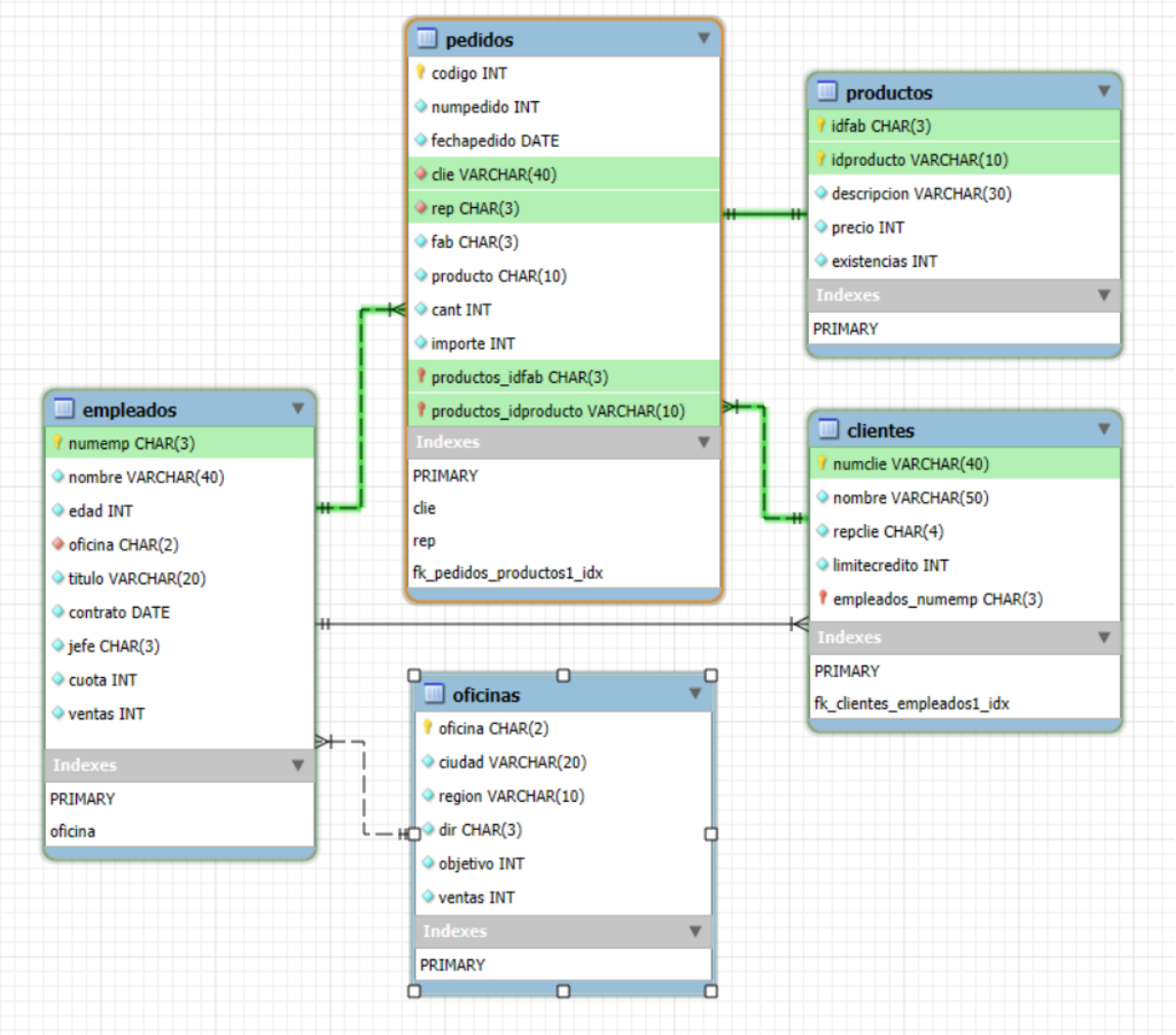
4. DISEÑO DEL MODELO ENTIDAD – RELACION (ER)

Entidades principales:

- **Clientes** (numclie): clave primaria.
- **Empleados** (numemp): clave primaria. Pueden ser representantes o directores.
- **Oficinas** (oficina): clave primaria.
- **Pedidos** (codigo): clave primaria, enlaza a clientes y representantes.
- **Productos** (idfab, idproducto): clave compuesta.

Relaciones:

- **Clientes — Empleados** (representantes): relación de muchos a uno (repclie → numemp).
- **Empleados — Oficinas**: muchos empleados trabajan en una oficina (oficina).
- **Pedidos — Clientes**: un cliente puede hacer muchos pedidos (clie → numclie).
- **Pedidos — Empleados**: un pedido está asociado a un representante (rep → numemp).
- **Pedidos — Productos**: pedido contiene un producto (fab, producto → idfab, idproducto).



Grafica 1. Modelo ER (Elaboración Propia)

Justificación del diseño:

- ✓ Está normalizado en **3FN (Tercera Forma Normal)**: no hay grupos repetitivos ni dependencias transitivas.
- ✓ Uso correcto de claves foráneas asegura integridad referencial:
  - No puedes insertar un pedido con un cliente o representante inexistente.
  - Cada producto en pedidos debe existir en productos.

5. CONSULTAS SQL

En este apartado nos encargaremos de realizar algunas consultas, desde las mas básicas tales como llamar y ver que contienen nuestras tablas, así como avanzadas que nos permitan extraer información valiosa para el análisis y recomendaciones prospectivas para el beneficio la empresa.

Consulta 1: Listar los nombres de los clientes que tienen asignado el representante Álvaro Jaumes.

```
SELECT nombre, repclie FROM clientes WHERE repclie = (SELECT numemp FROM empleados WHERE nombre = 'Alvaro Jaumes' )
```

[Editar en línea]

+ Opciones

nombre	repclie
Juan Suarez	102
Cristina Bulini	102
Juan Malo	102
Jose Libros	102

Explicación: se explora que clientes fueron atendidos por el representante en mención, asumiendo que no pueden existir representantes con el mismo nombre.

Consulta 2: Listar los vendedores (numemp, nombre, y nº de oficina) que trabajan en oficinas "buenas" donde sus ventas superan su objetivo.

```
SELECT numemp, nombre, oficina FROM empleados WHERE oficina = ANY (SELECT oficina FROM oficinas WHERE ventas > objetivo)
```

[Editar en línea]

+ Opciones

	numemp	nombre	oficina
Editar Copiar Borrar	102	Alvaro Jaumes	21
Editar Copiar Borrar	105	Vicente Pantalla	13
Editar Copiar Borrar	106	Luis Antonio	11
Editar Copiar Borrar	108	Ana Bustamante	21
Editar Copiar Borrar	109	Maria Sunta	11
Editar Copiar Borrar	110	Juan Victor	11

Explicación: Se extraen los empleados que trabajan en oficinas que tienen ventas superiores a sus objetivos.

Consulta 3: Listar los vendedores que no trabajan en oficinas dirigidas por el empleado 108.

```
SELECT numemp, nombre, oficina, jefe FROM empleados WHERE oficina NOT IN ( SELECT oficina FROM oficinas WHERE dir = 108)
```

[Editar en línea]

+ Opciones

	numemp	nombre	oficina	jefe
Editar Copiar Borrar	101	Antonio Viguer	12	104
Editar Copiar Borrar	103	Juan Rovira	12	104
Editar Copiar Borrar	104	Jose Gozalez	12	106
Editar Copiar Borrar	105	Vicente Pantalla	13	104
Editar Copiar Borrar	106	Luis Antonio	11	
Editar Copiar Borrar	109	Maria Sunta	11	106
Editar Copiar Borrar	110	Juan Victor	11	104

Explicación: Se observan vendedores que no se encuentran en la oficina dirigida por el jefe 108.

Consulta 4: Listar los productos (idfab, idproducto y descripción) para los cuales no se ha recibido ningún pedido de 25000 o más.

```
Query 1   ferreteria*   ferreteria - Schema
Limit to 1000 rows
1  use ferreteria;
2
3  select idfab, idproducto, descripcion from productos where exists(select * from pedidos where fab = idfab and producto = idproducto and importe <= 25000);
```

Result Grid | Filter Rows:

idfab	idproducto	descripcion
aci	4100z	mont
aci	41004	art t4
rei	2a44g	pas
fea	114	cubo
aci	41002	bisagra
bic	41003	manivela
inm	779c	reostato 3
inm	773c	reostato
aci	41003	art t3
qsa	xk47	red
rei	2a44r	bomba r
aci	4100x	junta
fea	112	cubo
NULL	NULL	NULL

Explicación: Se observan los productos donde su importe es menor a 25000.

Consulta 5: Listar las oficinas en donde todos los vendedores tienen ventas que superan al 50% del objetivo de la oficina.

```
SELECT * FROM oficinas WHERE (objetivo * .5) <= (SELECT MIN(ventas) FROM empleados WHERE empleados.oficina = oficinas.oficina)
```

[Editar en línea] [Editar]

+ Opciones

	oficina	ciudad	region	dir	objetivo	ventas
Editar Copiar Borrar	13	Castellon	este	105	350000	368000
Editar Copiar Borrar	22	A Coruña	oeste	108	300000	186000

Explicación: Vendedores que tienen ventas superiores al 50% del objetivo.

Consulta 6: Listar las oficinas en donde haya un vendedor cuyas ventas representen más del 55% del objetivo de su oficina.

```
SELECT * FROM oficinas WHERE EXISTS ( SELECT * FROM empleados WHERE empleados.oficina=oficinas.oficina AND ventas > objetivo * 0.55)
```

[Editar en línea] [Editar] [

+ Opciones

	oficina	ciudad	region	dir	objetivo	ventas
Editar Copiar Borrar	11	Valencia	este	106	575000	693000
Editar Copiar Borrar	13	Castellon	este	105	350000	368000
Editar Copiar Borrar	21	Babajoz	oeste	108	725000	836000
Editar Copiar Borrar	22	A Coruña	oeste	108	300000	186000

Explicación: Oficina con ventas mayores al 55% del objetivo.

Consulta 7: Listar los clientes asignados a Ana Bustamante que no han remitido un pedido superior a 3000 pts.

```
SELECT numclie, nombre FROM clientes WHERE repclie IN ( SELECT numemp FROM empleados WHERE nombre = 'Ana Bustamante' ) AND numclie NOT IN ( SELECT clie FROM pedidos WHERE
importe > 3000 AND clie IS NOT NULL)
```

[Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

+ Opciones

	numclie	nombre
Editar Copiar Borrar	2118	Junipero Alvarez

Explicación: Clientes asignados a un vendedor que no supera su pedido al valor determinado por la consulta.

Listar las oficinas que tengan un objetivo mayor que la suma de las cuotas de sus vendedores.

```
SELECT * FROM oficinas WHERE objetivo > ( SELECT SUM(cuota) FROM empleados WHERE empleados.oficina = oficinas.oficina)
```

[Editar en línea]

+ Opciones

	oficina	ciudad	region	dir	objetivo	ventas
Editar Copiar Borrar	12	Alicante	este	104	800000	735000
Editar Copiar Borrar	21	Babajoz	oeste	108	725000	836000

Consulta 8: Total de ventas por representante

SELECT e.numemp, e.nombre, SUM(p.importe) AS total_ventas

FROM empleados e

JOIN pedidos p ON e.numemp = p.rep

GROUP BY e.numemp, e.nombre

ORDER BY total_ventas DESC;

Explicación: Relacionamos empleados con pedidos, agrupando por empleado para sumar las ventas (importe). Se observa quién vende más.

Consulta 9: Clientes que han superado su límite de crédito

SELECT c.numclie, c.nombre, c.limitecredito, SUM(p.importe) AS total_pedidos

FROM clientes c

JOIN pedidos p ON c.numclie = p.clie

GROUP BY c.numclie, c.nombre, c.limitecredito

HAVING SUM(p.importe) > c.limitecredito;

Explicación: Identifica a clientes con pedidos cuyo total excede el crédito autorizado. Ayuda a detectar riesgos financieros.

Consulta 10: Oficinas con ventas menores a su objetivo

```
SELECT oficina, ciudad, ventas, objetivo
FROM oficinas
WHERE ventas < objetivo;
```

Explicación: Identifica oficinas que no han alcanzado su meta de ventas. Útil para evaluar desempeño regional.

Consulta 11: Productos más vendidos por cantidad

```
SELECT pr.descripcion, SUM(p.cant) AS cantidad_total
FROM pedidos p
JOIN productos pr ON p.fab = pr.idfab AND p.producto = pr.idproducto
GROUP BY pr.descripcion
ORDER BY cantidad_total DESC
LIMIT 5;
```

Explicación: Muestra los 5 productos con más unidades vendidas.

Consulta 12: Ingresos totales por producto

```
SELECT pr.descripcion, SUM(p.importe) AS ingresos
FROM pedidos p
JOIN productos pr ON p.fab = pr.idfab AND p.producto = pr.idproducto
GROUP BY pr.descripcion
ORDER BY ingresos DESC
LIMIT 5;
```

Explicación: Similar a la anterior, pero mide ingresos, no unidades.

Consulta 13: Total de ventas por representante

```
SELECT
    e.numemp,
    e.nombre,
    SUM(p.importe) AS total_ventas
FROM
    empleados e
JOIN
    pedidos p ON e.numemp = p.rep
GROUP BY
    e.numemp, e.nombre
ORDER BY
    total_ventas DESC;
```

Explicación: Muestra cuánto ha vendido cada representante (empleado), sumando el importe de sus pedidos. Útil para evaluar rendimiento comercial.

Consulta 14: Productos más vendidos (por cantidad)

```
SELECT
    pr.descripcion,
    SUM(pe.cant) AS total_cantidad_vendida
FROM
    pedidos pe
JOIN
```

```
    productos pr ON pe.fab = pr.idfab AND pe.producto = pr.idproducto
GROUP BY
    pr.descripcion
ORDER BY
    total_cantidad_vendida DESC
LIMIT 5;
```

Explicación: Identifica los 5 productos con mayor número de unidades vendidas. Útil para decisiones de stock y promociones.

Consulta 15: Clientes con mayor volumen de compras

```
SELECT
    c.numclie,
    c.nombre,
    SUM(p.importe) AS total_compras
FROM
    clientes c
JOIN
    pedidos p ON c.numclie = p.clie
GROUP BY
    c.numclie, c.nombre
ORDER BY
    total_compras DESC
LIMIT 10;
```

Explicación: Permite identificar los clientes más valiosos para la empresa. Ideal para programas de fidelización.

Consulta 16: Comparar ventas vs. objetivo por oficina

```
SELECT
    o.oficina,
    o.ciudad,
    o.objetivo,
    o.ventas,
    (o.ventas - o.objetivo) AS diferencia
FROM
    oficinas o
ORDER BY
    diferencia DESC;
```

Explicación: Ayuda a evaluar el desempeño por oficina. Si la diferencia es negativa, significa que no alcanzaron el objetivo.

Consulta 17: Pedidos con productos sin existencias

```
SELECT
    p.codigo,
    p.numpedido,
    pr.descripcion,
    pr.existencias
FROM
```

```
    pedidos p
JOIN
    productos pr ON p.fab = pr.idfab AND p.producto = pr.idproducto
WHERE
    pr.existencias = 0;
```

Explicación: Muestra pedidos asociados a productos que actualmente no tienen existencias. Es vital para prever rupturas de stock.

Consulta 18: Jerarquía de empleados (subordinados por jefe)

```
SELECT
    jefe.nombre AS jefe,
    emp.nombre AS subordinado
FROM
    empleados emp
JOIN
    empleados jefe ON emp.jefe = jefe.numemp
ORDER BY
    jefe.nombre;
```

Explicación: Muestra qué empleados reportan a quién. Útil para visualizar la jerarquía interna y relaciones laborales.

Consulta 19: Pedidos por mes y año (análisis temporal)

```
SELECT
    YEAR(fechapedido) AS año,
```

```
    MONTH(fechapedido) AS mes,
    COUNT(*) AS total_pedidos,
    SUM(importe) AS total_importe
FROM
    pedidos
GROUP BY
    YEAR(fechapedido), MONTH(fechapedido)
ORDER BY
    año, mes;
```

Explicación: Permite observar tendencias temporales de ventas. Ideal para previsiones y análisis estacional.

Consulta 20: Clientes sin pedidos

```
SELECT
    c.numclie,
    c.nombre
FROM
    clientes c
LEFT JOIN
    pedidos p ON c.numclie = p.clie
WHERE
    p.codigo IS NULL;
```

Explicación: Detecta clientes registrados que aún no han realizado ningún pedido. Pueden ser objeto de campañas de activación.

6. RECOMENDACIONES ESTRATÉGICAS

1. Optimizar la distribución de metas comerciales: Realizar un análisis trimestral de cumplimiento por oficina y ajustar los objetivos según el potencial real de cada zona. Incentivar con bonos a los representantes que superen sus metas y dar seguimiento a quienes están por debajo.

2. Implementar un sistema de gestión de inventario predictivo: Integrar alertas automáticas cuando el stock de un producto descienda a niveles críticos. Priorizar la reposición de productos con alta rotación y eliminar aquellos que no se venden o están obsoletos.

3. Desarrollar programas de fidelización para clientes inactivos: Crear campañas específicas (descuentos, promociones) para reactivar clientes sin pedidos recientes. Analizar causas de inactividad (precios, productos, atención) y proponer soluciones específicas por segmento.

4. Evaluar desempeño individual con métricas claras: Llevar un seguimiento mensual de ventas por representante, cantidad de pedidos y nuevos clientes generados. Invertir en capacitación para mejorar el rendimiento de los empleados con menor desempeño.

5. Mejorar los procesos de toma de pedidos: Validar en tiempo real el stock disponible antes de confirmar un pedido para evitar ventas sin existencias. Establecer un sistema que impida confirmar pedidos de productos agotados sin una fecha de reposición clara.

6. Validar valores antes de insertar: Aplicar constraints adicionales o validaciones para evitar errores en campos como fechas, cantidades negativas o límites de crédito incorrectos.

7. Diseñar vistas para análisis ejecutivo: Crear vistas como ventas_por_empleado, clientes_con_mas_pedidos o productos_top_ventas para facilitar el análisis.

7. ANÁLISIS EN PROSPECTIVA

A medida que la ferretería crece y diversifica su catálogo, se vuelve indispensable profesionalizar la gestión operativa y comercial. Aquí detallamos algunas líneas de acción con visión a futuro:

1. Automatización y análisis predictivo:

Utilizando los datos históricos de ventas, pedidos y objetivos por oficina, se puede aplicar inteligencia de negocio (BI) para predecir comportamientos de compra, estacionalidad y ciclos de ventas, desempeño de empleados. Integrando herramientas de analítica avanzada como Power BI, Tableau o Python con Pandas para identificar patrones de éxito y riesgo.

2. Integración con e-commerce: El modelo permite fácilmente incorporar un sistema de ventas en línea, añadiendo una entidad como usuarios_web y carrito_compras, sin afectar la estructura actual.

3. Expansión geográfica o virtual:

Las oficinas que muestran ventas exitosas podrían ser modelo para abrir nuevas sucursales en zonas similares teniendo en cuenta que ya la base de datos está segmentada por región (este, norte, centro, oeste). Evaluar una estrategia de ventas en línea para ampliar el alcance geográfico sin necesidad de expansión física.

4. Transformación digital de procesos

Migrar hacia un sistema ERP que centralice la información de clientes, ventas, inventarios y empleados en tiempo real. Incorporar métricas de satisfacción del cliente (NPS) y tiempo de entrega para mejorar la experiencia general.

5. Segmentación avanzada de clientes

Implementar un CRM que clasifique a los clientes por historial, frecuencia y valor de compra para personalizar campañas. Aplicar modelos de segmentación como RFM (Recency, Frequency, Monetary) para dirigir recursos de forma efectiva.

6. Seguridad y acceso: Es recomendable implementar roles de acceso para limitar operaciones (lectura, inserción, eliminación) dependiendo del perfil de usuario: administrador, vendedor, supervisor.

8. CONCLUSIONES

El análisis exhaustivo de la base de datos revela que la ferretería cuenta con una estructura relacional bien normalizada, donde se respetan los principios clave de la **integridad referencial** y la **consistencia de datos**. La empresa ha mantenido un registro ordenado de clientes, empleados, productos, oficinas y pedidos, lo que permite un seguimiento preciso de las operaciones comerciales, desde las ventas por representante hasta la disponibilidad de inventario.

Los datos reflejan una organización jerárquica clara, con empleados vinculados a oficinas y representantes de ventas asociados a clientes y pedidos. Además, la implementación de claves foráneas con restricciones ON DELETE CASCADE y ON UPDATE CASCADE mejora la robustez del sistema frente a cambios o eliminaciones de registros relacionados.

A través de las consultas desarrolladas, se evidencian varios aspectos críticos:

- **Concentración de ventas**: Un reducido grupo de representantes acumula la mayoría de las ventas, lo que genera dependencia de ciertos empleados.
- **Desbalance entre objetivo y ventas por oficina**: Algunas oficinas sobrepasan el objetivo, mientras otras están por debajo de su meta.
- **Falta de existencias en productos populares**: Se han realizado pedidos de productos que ya no tienen stock, indicando carencias en la planificación de inventarios.
- **Clientes inactivos**: Existen clientes registrados sin actividad de compra, lo que representa una oportunidad desaprovechada.
- Se evidencia una **interrelación eficiente** entre ventas, productos, oficinas y personal.

9. BIBLIOGRAFIA

MySQL, Que es y como se usa (Jeffrey Erickson, 2024, agosto 29).
https://www.oracle.com/co/mysql/what-is-mysql/