

Asignatura	Datos del alumno	Fecha
Bases De Datos	Apellidos: De Mendoza	9/11/2024
	Nombre: Alejandro	

ACTIVIDAD LABORATORIO NO.1  
LABORATORIO: PROGRAMACIÓN DE CONSULTAS COMPLEJAS Y VISTAS EN SQL

PRESENTADO POR:  
ALEJANDRO DE MENDOZA

PRESENTADO AL PROFESOR:  
ING JAVIER DIAZ DIAZ

FUNDACIÓN UNIVERSITARIA INTERNACIONAL DE LA RIOJA  
BOGOTÁ D.C.  
9 DE NOVIEMBRE  
2024

Asignatura	Datos del alumno	Fecha
Bases De Datos	Apellidos: De Mendoza	9/11/2024
	Nombre: Alejandro	

## TABLA DE CONTENIDO

INTRODUCCIÓN .....	3
DESARROLLO DEL LABORATORIO.....	3
GENERAR SENTENCIAS DDL.....	3
CONSULTAS DE INSERCIÓN .....	6
EJECUCIÓN DE CONSULTAS.....	6
PRIMERA CONSULTA .....	6
SEGUNDA CONSULTA .....	7
TERCERA CONSULTA.....	7
CUARTA CONSULTA.....	8
QUINTA CONSULTA .....	8
VISTA RESUMEN PEDIDOS.....	9
CONSULTA DE VISTA RESUMEN PEDIDOS .....	9
CONCLUSIÓN .....	10
BIBLIOGRAFÍA .....	10

Asignatura	Datos del alumno	Fecha
Bases De Datos	Apellidos: De Mendoza	9/11/2024
	Nombre: Alejandro	

## INTRODUCCIÓN

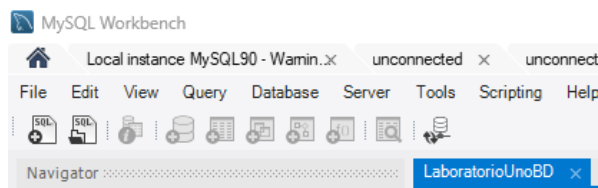
Como introducción y para la realización de este laboratorio se estudiarán los temas sobre SQL: introducción y estructura básica y SQL: operaciones y funciones. Y es por esto que este desarrollo de este laboratorio tiene como objetivo que se ponga en práctica lo estudiado en los temas relacionados del aula con SQL, que son los siguientes:

- ▶ Creación de tablas.
- ▶ Realizar inserciones.
- ▶ Utilización de consultas con SELECT.
- ▶ Operaciones sobre conjuntos.
- ▶ Funciones de agregación.
- ▶ Consultas complejas.
- ▶ Vistas.

Considero importante indicar en esta introducción que para este desarrollo nos vamos a basar en la plataforma MySQL que es un sistema de gestión de bases de datos relacional (RDBMS) de código abierto utilizado para almacenar y administrar datos estructurados.

## DESARROLLO DEL LABORATORIO

Para este desarrollo lo primero es crear el proyecto en SQL que para este caso lo nombre LaboratorioUnoBD. A continuación, la imagen respectiva en MySQL:



## GENERAR SENTENCIAS DDL

Procedo entonces a revisar las tablas que se deben construir, para este caso son 3, y determino que UNIR CLIENTE no depende de ninguna tabla entonces es la primera tabla a crear, además UNIR COMERCIAL tampoco depende de ninguna tabla por lo que podría ser la segunda tabla. Es importante precisar que no puedo crear UNIR PEDIDO porque necesito que estén creadas las tablas de UNIR COMERCIAL y UNIR CLIENTE, es decir, primero se crean las tablas que no tengan dependencia de otras.

**NOTA IMPORTANTE:** Las tablas están nombradas con separadores de ".". En los nombres aparece UNIR.COMERCIAL, y si uno genera el nombre de la tabla UNIR.COMERCIAL aparece el siguiente error en MySQL: *Error Code: 1049. Unknown database 'unir'*, es por esto que se va a reemplazar en este laboratorio el punto ".", por un guion bajo "\_" para designar los nombres de las tablas, para de esta forma UNIR.COMERCIAL ser reemplazado por UNIR\_COMERCIAL y de igual manera para el resto de tablas.

Ahora lo siguiente en este caso es crear la base de datos que la nombre como LaboratorioUnoBasesDeDatos. Entonces para crear las tablas, la primera sentencia que voy a trabajar es el create table para luego definir el nombre de la tabla, definiendo sus campos y características de cada uno de ellos indicando los tipos de datos y sus longitudes. Además de eso identificando las restricciones que tienen los campos con el constraint que es la palabra reservada que significa restricción.

Continuando en el proceso de creación de la tabla defino el nombre de la tabla y para esto utilizo la sentencia create table + nombre de la tabla. La tabla en este caso se llama "UNIR CLIENTE", entonces utilizo la sentencia create table UNIR\_CLIENTE, luego entre paréntesis incluyo la lista de los campos. Ahora, como se puede ver hay campos que son alfanuméricos por lo que para estos se designa la palabra reservada varchar y entre paréntesis la longitud de cada campo indicando que no son campos vacíos o nulos. A continuación, la imagen:

Asignatura	Datos del alumno	Fecha
Bases De Datos	Apellidos: De Mendoza	9/11/2024
	Nombre: Alejandro	

```

4      -- Tabla UNIR CLIENTE
5      CREATE TABLE UNIR_CLIENTE (
6          ID int not null,
7          NOMBRE varchar(100) not null,
8          APELLIDO1 varchar(100) not null,
9          APELLIDO2 varchar(100),
10         CIUDAD varchar (100),
11         CATEGORIA int,

```

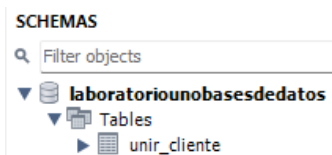
Procedo entonces a colocar las restricciones que controlan la integridad referencial y para este caso solo hay una restricción que es de llave primaria. A continuación, la imagen respectiva:

```

4      -- Tabla UNIR CLIENTE
5      CREATE TABLE UNIR_CLIENTE (
6          ID int not null,
7          NOMBRE varchar(100) not null,
8          APELLIDO1 varchar(100) not null,
9          APELLIDO2 varchar(100),
10         CIUDAD varchar (100),
11         CATEGORIA int,
12         CONSTRAINT CLIENTE_PK PRIMARY KEY (ID)
13     );

```

Ahora, actualizando en la parte izquierda los esquemas nos muestra la tabla de nombre "UNIR CLIENTE":



Reviso la tabla ejecutando la sentencia **describe** + el nombre de la tabla, y nos muestra la estructura de la tabla, en donde se pueden observar los campos, el tipo de datos, valores nulos y la restricción PK:

```
15 describe UNIR_CLIENTE;
```

Field	Type	Null	Key	Default	Extra
ID	int	NO	PRI	NULL	
NOMBRE	varchar(100)	NO		NULL	
APELLIDO1	varchar(100)	NO		NULL	
APELLIDO2	varchar(100)	YES		NULL	
CIUDAD	varchar(100)	YES		NULL	
CATEGORIA	int	YES		NULL	

Ahora hacemos lo mismo con la tabla UNIR COMERCIAL. A continuación, la imagen de la creación:

```

17      -- Tabla UNIR COMERCIAL
18      CREATE TABLE UNIR_COMERCIAL (
19          ID INT NOT NULL,
20          NOMBRE VARCHAR(100) NOT NULL,
21          APELLIDO1 VARCHAR(100) NOT NULL,
22          APELLIDO2 VARCHAR(100),
23          COMISION DECIMAL(10, 2),
24          CONSTRAINT COMERCIAL_PK PRIMARY KEY (ID)
25     );

```

```
27 describe UNIR_COMERCIAL;
```

Field	Type	Null	Key	Default	Extra
ID	int	NO	PRI	NULL	
NOMBRE	varchar(100)	NO		NULL	
APELLIDO1	varchar(100)	NO		NULL	
APELLIDO2	varchar(100)	YES		NULL	
COMISION	decimal(10,2)	YES		NULL	

Teniendo las dos tablas padres, creo la tabla **hija** que va a ser la tabla **UNIR PEDIDO** indicando que en esta se incluyen dos **llaves foráneas** donde la restricción de cada FK define la relación con las entidades UNIR COMERCIAL y UNIR CLIENTE. El desarrollo queda de la siguiente manera:

Asignatura	Datos del alumno	Fecha
Bases De Datos	Apellidos: De Mendoza	9/11/2024
	Nombre: Alejandro	

```

29 -- Tabla UNIR_PEDIDO
30 CREATE TABLE UNIR_PEDIDO (
31     ID INT NOT NULL,
32     TOTAL DECIMAL(10, 2) NOT NULL,
33     FECHA DATE,
34     ID_CLIENTE INT NOT NULL,
35     ID_COMERCIAL INT NOT NULL,
36     CONSTRAINT PEDIDO_PK PRIMARY KEY (ID),
37     CONSTRAINT FK_CLIENTE FOREIGN KEY (ID_CLIENTE) REFERENCES UNIR_CLIENTE(ID),
38     CONSTRAINT FK_COMERCIAL FOREIGN KEY (ID_COMERCIAL) REFERENCES UNIR_COMERCIAL(ID)
39 );

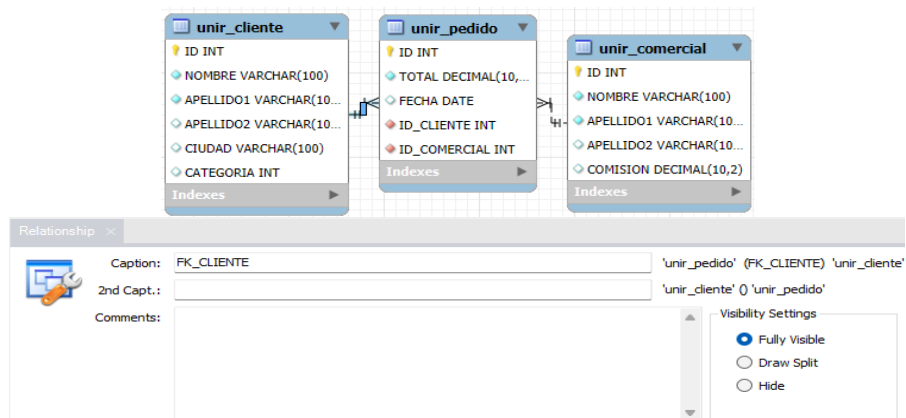
```

41 describe UNIR\_PEDIDO;

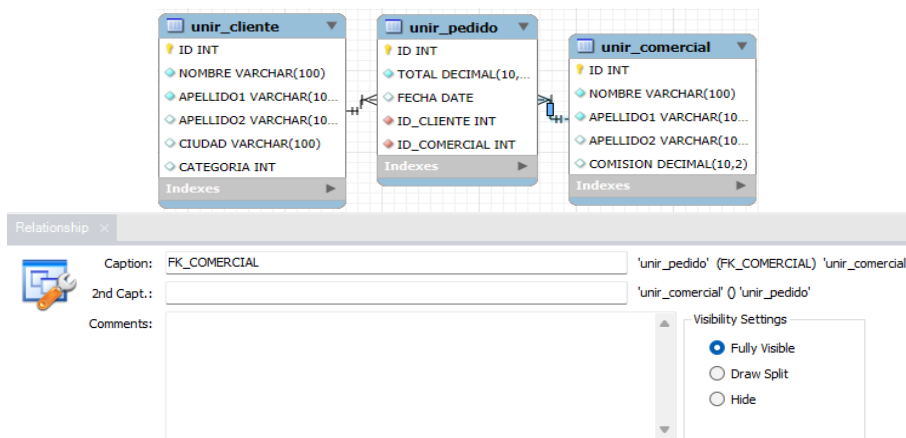
42



Field	Type	Null	Key	Default	Extra
ID	int	NO	PRI	NULL	
TOTAL	decimal(10,2)	NO	YES	NULL	
FECHA	date	YES		NULL	
ID_CLIENTE	int	NO	MUL	NULL	
ID_COMERCIAL	int	NO	MUL	NULL	

Explicando, coloco el foreign key, luego entre paréntesis el campo de la tabla en la que estoy trabajando que es la llave foránea, que es **ID\_CLIENTE** para la tabla UNIR\_CLIENTE, y **ID\_COMERCIAL** para la tabla UNIR\_COMERCIAL, y las referencio con las tablas padre UNIR\_CLIENTE y UNIR\_COMERCIAL con la palabra reservada **reference** colocando el nombre de las tablas y entre paréntesis el nombre de la llave primaria con la que desarrollo la relación que en este caso es el campo ID. **NOTA:** Aclaro que la longitud y tipo de dato de ID\_CLIENTE y ID\_COMERCIAL deben ser exactamente iguales tanto en las tablas padre como en la tabla hija. Se evidencia que el campo ID es una llave primaria, no recibe nulos, por otra parte, ID\_CLIENTE y ID\_COMERCIAL son llaves múltiples que representan llaves foráneas. Ahora teniendo la construcción de la estructura completa, agregamos un diagrama para determinar la relación en esquemas mostrando la relación entre las dos tablas para ser más específicos FK\_CLIENTE de la tabla UNIR\_CLIENTE, a continuación, las imágenes respectivas:



Y la relación FK\_COMERCIAL de la tabla UNIR\_COMERCIAL:



**NOTA:** Este símbolo  denota es la tabla padre, y las líneas abiertas indican que es la tabla hija , por ende, estas son unas relaciones de uno a muchos. Y lo correcto es que un cliente tenga muchos pedidos, y un vendedor tenga de igual manera muchos pedidos, entonces esta correcto el esquema.

Asignatura	Datos del alumno	Fecha
Bases De Datos	Apellidos: De Mendoza	9/11/2024
	Nombre: Alejandro	

## CONSULTAS DE INSERCIÓN

Como ya tenemos creada la estructura de todas las tablas y sus relaciones, procedemos al ingreso de sus datos. Para esto, ingreso primero los datos de las tablas padre UNIR\_CLIENTE y UNIR\_COMERCIAL antes de insertar los datos en UNIR\_PEDIDO, ya que esta tabla depende de las dos primeras a través de llaves foráneas. Entonces utilizo las palabras reservadas **INSERT INTO** + nombre de la tabla + (nombres de campos donde vamos a insertar los valores). Para el caso de la tabla UNIR\_CLIENTE el código y el resultado al ejecutar dando **describe** en la tabla, quedan:

-- Inserción en UNIR\_CLIENTE
INSERT INTO UNIR\_CLIENTE (ID, NOMBRE, APELLIDO1, APELLIDO2, CIUDAD, CATEGORIA) VALUES
(1, 'Aarón', 'Rívero', 'Gómez', 'Almería', 100),
(2, 'Adela', 'Salas', 'Díaz', 'Granada', 200),
(3, 'Adolfo', 'Rubio', 'Flores', 'Sevilla', NULL),
(4, 'Adrián', 'Suárez', NULL, 'Jaén', 300),
(5, 'Marcos', 'Loyola', 'Méndez', 'Almería', 200),
(6, 'María', 'Santana', 'Moreno', 'Cádiz', 100),
(7, 'Pilar', 'Ruiz', NULL, 'Sevilla', 300),
(8, 'Pepe', 'Ruiz', 'Santana', 'Huelva', 200),
(9, 'Guillermo', 'López', 'Gómez', 'Granada', 225),
(10, 'Daniel', 'Santana', 'Loyola', 'Sevilla', 125);

ID	NOMBRE	APELLIDO1	APELLIDO2	CIUDAD	CATEGORIA
1	Aarón	Rívero	Gómez	Almería	100
2	Adela	Salas	Díaz	Granada	200
3	Adolfo	Rubio	Flores	Sevilla	NULL
4	Adrián	Suárez	NULL	Jaén	300
5	Marcos	Loyola	Méndez	Almería	200
6	María	Santana	Moreno	Cádiz	100
7	Pilar	Ruiz	NULL	Sevilla	300
8	Pepe	Ruiz	Santana	Huelva	200
9	Guillermo	López	Gómez	Granada	225
10	Daniel	Santana	Loyola	Sevilla	125
NULL	NULL	NULL	NULL	NULL	NULL

Ahora hacemos lo mismo con la tabla UNIR\_COMERCIAL, y la inserción queda de la siguiente manera:

```
-- Inserción en UNIR_COMERCIAL
INSERT INTO UNIR_COMERCIAL (ID, NOMBRE, APELLIDO1, APELLIDO2, COMISION) VALUES
(1, 'Daniel', 'Sáez', 'Vega', 0.15),
(2, 'Juan', 'Gómez', 'López', 0.13),
(3, 'Diego', 'Flores', 'Salas', 0.11),
(4, 'Marta', 'Herrera', 'Gil', 0.14),
(5, 'Antonio', 'Carretero', 'Ortega', 0.12),
(6, 'Manuel', 'Dominguez', 'Hernández', 0.13),
(7, 'Antonio', 'Vega', 'Hernández', 0.11),
(8, 'Alfredo', 'Ruiz', 'Flores', 0.05);
```

ID	NOMBRE	APELLIDO1	APELLIDO2	COMISION
1	Daniel	Sáez	Vega	0.15
2	Juan	Gómez	López	0.13
3	Diego	Flores	Salas	0.11
4	Marta	Herrera	Gil	0.14
5	Antonio	Carretero	Ortega	0.12
6	Manuel	Dominguez	Hernández	0.13
7	Antonio	Vega	Hernández	0.11
8	Alfredo	Ruiz	Flores	0.05
NULL	NULL	NULL	NULL	NULL

Y por último hago la inserción de datos en la tabla UNIR\_PEDIDO, finalizando:

-- Inserción en UNIR_PEDIDO									
INSERT INTO UNIR_PEDIDO (ID, TOTAL, FECHA, ID_CLIENTE, ID_COMERCIAL) VALUES					ID	TOTAL	FECHA	ID_CLIENTE	ID_COMERCIAL
(1, 150.5, '2017-10-05', 5, 2),					1	150.50	2017-10-05	5	2
(2, 270.65, '2016-09-10', 1, 5),					2	270.65	2016-09-10	1	5
(3, 65.26, '2017-10-05', 2, 1),					3	65.26	2017-10-05	2	1
(4, 110.5, '2016-08-17', 8, 3),					4	110.50	2016-08-17	8	3
(5, 948.5, '2017-09-10', 5, 2),					5	948.50	2017-09-10	5	2
(6, 2400.6, '2016-07-27', 7, 1),					6	2400.60	2016-07-27	7	1
(7, 5760, '2015-09-10', 2, 1),					7	5760.00	2015-09-10	2	1
(8, 1983.43, '2017-10-10', 4, 6),					8	1983.43	2017-10-10	4	6
(9, 2480.4, '2016-10-10', 8, 3),					9	2480.40	2016-10-10	8	3
(10, 250.45, '2015-06-27', 8, 2),					10	250.45	2015-06-27	8	2
(11, 75.29, '2016-08-17', 3, 7),					11	75.29	2016-08-17	3	7
(12, 3045.6, '2017-04-25', 2, 1),					12	3045.60	2017-04-25	2	1
(13, 545.75, '2019-01-25', 6, 1),					13	545.75	2019-01-25	6	1
(14, 145.82, '2017-02-02', 6, 1),					14	145.82	2017-02-02	6	1
(15, 370.85, '2019-03-11', 1, 5),					15	370.85	2019-03-11	1	5
(16, 2389.23, '2019-03-11', 1, 5);					16	2389.23	2019-03-11	1	5
					NULL	NULL	NULL	NULL	NULL

## EJECUCIÓN DE CONSULTAS

Ahora como ya tenemos creada toda la estructura de todas las tablas, sus relaciones y sus datos ingresados, entonces procedo a ejecutar el proceso de consultas de la actividad.

### PRIMERA CONSULTA

Selecciono todos los campos de la tabla UNIR\_PEDIDO y ordeno los resultados por la columna FECHA en orden descendente, lo que permite que los pedidos más recientes aparezcan en la parte superior del listado y para esto ejecuto la siguiente sentencia:

Asignatura	Datos del alumno	Fecha
Bases De Datos	Apellidos: De Mendoza	9/11/2024
	Nombre: Alejandro	

```

87 • SELECT *
88 FROM UNIR_PEDIDO
89 ORDER BY FECHA DESC;

```

Donde: **SELECT \*:** Selecciona todos los campos. **FROM UNIR\_PEDIDO:** Desde la tabla UNIR\_PEDIDO. **ORDER BY FECHA DESC:** Ordena por fecha en orden descendente. Ejecuto y trae como resultado:

	ID	TOTAL	FECHA	ID_CLIENTE	ID_COMERCIAL
▶	15	370.85	2019-03-11	1	5
	16	2389.23	2019-03-11	1	5
	13	545.75	2019-01-25	6	1
	8	1983.43	2017-10-10	4	6
	1	150.50	2017-10-05	5	2
	3	65.26	2017-10-05	2	1
	5	948.50	2017-09-10	5	2
	12	3045.60	2017-04-25	2	1
	14	145.82	2017-02-02	6	1
	9	2480.40	2016-10-10	8	3
	2	270.65	2016-09-10	1	5
	4	110.50	2016-08-17	8	3
	11	75.29	2016-08-17	3	7
	6	2400.60	2016-07-27	7	1
	7	5760.00	2015-09-10	2	1
	10	250.45	2015-06-27	8	2
•	NULL	NULL	NULL	NULL	NULL

Vemos que nos trae los pedidos realizados filtrados por fecha de realización en orden descendente (mostrando primero los más recientes) por lo que la consulta esta correcta.

## SEGUNDA CONSULTA

Selecciono todos los campos de la tabla UNIR\_PEDIDO, filtro los resultados para que solo se muestren los pedidos con un TOTAL entre 300 y 600 y ordeno los resultados por la columna FECHA en orden ascendente, ordenando los pedidos desde los más antiguos a los más recientes y ejecuto:

```

91 • SELECT *
92 FROM UNIR_PEDIDO
93 WHERE TOTAL BETWEEN 300 AND 600
94 ORDER BY FECHA ASC;

```

Donde: **SELECT \*:** Selecciona todos los campos de una tabla. **FROM UNIR\_PEDIDO:** Desde la tabla UNIR\_PEDIDO. **WHERE TOTAL BETWEEN 300 AND 600:** Filtra los resultados para que solo se muestren los pedidos con un TOTAL entre 300 y 600. **ORDER BY FECHA ASC:** Ordena por fecha en orden ascendente. El resultado entonces es:

	ID	TOTAL	FECHA	ID_CLIENTE	ID_COMERCIAL
▶	13	545.75	2019-01-25	6	1
	15	370.85	2019-03-11	1	5
	NULL	NULL	NULL	NULL	NULL

Como se denota, la cantidad total se encuentra entre 300 y 600 y esta ordenado por fecha de manera ascendente (mostrando primero los más antiguos, a la inversa de un orden descendente), esta correcta la consulta.

## TERCERA CONSULTA

Selecciono los campos ID, NOMBRE y APELLIDO1 de la tabla UNIR\_CLIENTE, filtro los resultados para que solo se muestren los clientes cuyo segundo apellido es NULL, ordeno los resultados por la columna APELLIDO1 y por la columna NOMBRE en orden ascendente y ejecuto:

```

96 • SELECT ID, NOMBRE, APELLIDO1
97 FROM UNIR_CLIENTE
98 WHERE APELLIDO2 IS NULL
99 ORDER BY APELLIDO1 ASC, NOMBRE ASC;

```

Donde: **SELECT ID, NOMBRE, APELLIDO1:** Selecciona los campos ID, NOMBRE y APELLIDO1. **FROM UNIR\_CLIENTE:** Desde tabla UNIR\_CLIENTE. **WHERE APELLIDO2 IS NULL:** Donde el segundo apellido este vacío. **ORDER BY APELLIDO1 ASC, NOMBRE ASC:** Ordena por el primer apellido y nombre de forma ascendente. El resultado es:

Asignatura	Datos del alumno	Fecha
Bases De Datos	Apellidos: De Mendoza	9/11/2024
	Nombre: Alejandro	

	ID	NOMBRE	APELLIDO1
▶	7	Pilar	Ruiz
	4	Adrián	Suárez
▲	NULL	NULL	NULL

Cómo podemos ver solo se muestran dos clientes cuyos campos son Null en sus segundos apellidos, además la consulta esta ordenada alfabéticamente por apellidos y luego nombres. Por lo que está correcto.

#### CUARTA CONSULTA

Selecciono los campos ID, NOMBRE, APELLIDO1 y APELLIDO2 de la tabla UNIR\_CLIENTE, filtro los resultados para que solo se muestren los clientes que han realizado un pedido con INNER JOIN y ordeno los resultados alfabéticamente eliminando los resultados repetidos con la cláusula DISTINCT y ejecuto.

```

101 • SELECT DISTINCT c.ID, c.NOMBRE, c.APELLIDO1, c.APELLIDO2
102 FROM UNIR_CLIENTE c
103 INNER JOIN UNIR_PEDIDO p ON c.ID = p.ID_CLIENTE
104 ORDER BY c.APELLIDO1 ASC, c.APELLIDO2 ASC, c.NOMBRE ASC;

```

Donde: **c**: Es un alias para compactar y brindar claridad, como diminutivo de cliente. **p**: Otro alias y uso "p" como diminutivo de pedido. **SELECT DISTINCT c.ID, c.NOMBRE, c.APELLIDO1, c.APELLIDO2**: Accede a las columnas de la tabla UNIR\_CLIENTE. Uso DISTINCT para eliminar cualquier registro duplicado. **FROM UNIR\_CLIENTE c**: Desde la tabla UNIR\_CLIENTE. **INNER JOIN UNIR\_PEDIDO p ON c.ID = p.ID\_CLIENTE**: Para esta sentencia INNER JOIN me permite la combinación de las tuplas de las dos tablas UNIR\_CLIENTE y UNIR\_PEDIDO, siempre y cuando cumplan una condición especificada en la cláusula **ON**. **INNER JOIN** devuelve solo las filas donde hay coincidencias en ambas tablas. Y **ON c.ID = p.ID\_CLIENTE** es la condición o la restricción de la combinación entre las dos tablas especificando que el INNER JOIN debe realizarse cuando el valor de ID en la tabla UNIR\_CLIENTE coincida con el valor de ID\_CLIENTE en la tabla UNIR\_PEDIDO. El resultado entonces es:

	ID	NOMBRE	APELLIDO1	APELLIDO2
▶	5	Marcos	Loyola	Méndez
	1	Aarón	Rivero	Gómez
	3	Adolfo	Rubio	Flores
	7	Pilar	Ruiz	NULL
	8	Pepe	Ruiz	Santana
	2	Adela	Salas	Díaz
	6	María	Santana	Moreno
	4	Adrián	Suárez	NULL

El resultado nos devuelve la tabla con el ID, el nombre, y los apellidos de todos los clientes que han realizado algún pedido. El listado esta ordenado alfabéticamente por primer apellido, segundo apellido y el nombre. No hay repetidos.

#### QUINTA CONSULTA

Selecciono los campos con alias com. (agregue los alias com., de comercial, para compactar y dar claridad), para la tabla UNIR\_COMERCIAL de la siguiente manera com.NOMBRE, com.APELLIDO1 y com.APELLIDO2, luego, filtro los resultados para que solo se muestren las personas comerciales que han participado en la realización de un pedido de la señora MARIA SANTANA MORENO y elimino los resultados repetidos utilizando la cláusula DISTINCT y ejecuto:

```

106 • SELECT DISTINCT com.NOMBRE, com.APELLIDO1, com.APELLIDO2
107 FROM UNIR_COMERCIAL com
108 INNER JOIN UNIR_PEDIDO p ON com.ID = p.ID_COMERCIAL
109 INNER JOIN UNIR_CLIENTE c1 ON p.ID_CLIENTE = c1.ID
110 WHERE c1.NOMBRE = 'María'
111 AND c1.APELLIDO1 = 'Santana'
112 AND c1.APELLIDO2 = 'Moreno';

```

Donde: **SELECT DISTINCT com.NOMBRE, com.APELLIDO1, com.APELLIDO2**: Elimina cualquier registro duplicado. **FROM UNIR\_COMERCIAL com**: Desde la tabla UNIR\_COMERCIAL. Se realiza un **INNER JOIN** entre UNIR\_PEDIDO y UNIR\_COMERCIAL para obtener los comerciales que se relacionan con los pedidos. Se realiza otro **INNER JOIN** entre UNIR\_CLIENTE y UNIR\_PEDIDO para asegurar que se están considerando solo los pedidos realizados por el cliente especificado en la cláusula WHERE que filtra el resultado para incluir únicamente los pedidos donde el cliente tiene los datos NOMBRE = 'María', APELLIDO1 = 'Santana', y APELLIDO2 = 'Moreno'. El resultado entonces es:



Asignatura	Datos del alumno	Fecha
Bases De Datos	Apellidos: De Mendoza	9/11/2024
	Nombre: Alejandro	

	NOMBRE	APELLIDO1	APELLIDO2
►	Daniel	Sáez	Vega

Se determina que el comercial que participo en algún pedido realizado por María Santana Moreno, señor Daniel Sáez Vega, pero solo se muestra una vez en la consulta ya que se eliminaron los resultados repetidos.

## VISTA RESUMEN PEDIDOS

Para la creación de la vista **ResumenPedidos** ejecuto una consulta que combine las tres tablas utilizando INNER JOIN para obtener los datos relevantes. En este caso la vista incluye, el ID del pedido, el nombre del cliente, el nombre del comercial, la fecha del pedido y el total. Para esto utilizo las palabras reservadas **CREATE VIEW** + el nombre de la vista **ResumenPedidos** + la palabra reservada **AS** que vincula el nombre de la vista con la consulta a ejecutar + la palabra reservada **SELECT** que es la definición de la vista, para luego incluir los campos y las formas de visualización de los mismos, y así incluir la palabra reservada **FROM** + el nombre de la tabla de donde provienen los datos, finalizando con dos **INNER JOIN** en este caso de la tabla PEDIDO con los registros de la tabla CLIENTE donde los ID\_CLIENTE coinciden. Y un INNER JOIN que une los registros de la tabla PEDIDO con los registros de la tabla COMERCIAL donde los ID\_COMERCIAL coinciden. Entonces la ejecución queda de la siguiente manera:

```

114 • CREATE VIEW ResumenPedidos AS
115 SELECT
116     p.ID AS ID_Pedido,
117     p.FECHA AS Fecha_Pedido,
118     cl.NOMBRE AS Nombre_Cliente,
119     cl.APELLIDO1 AS Apellido1_Cliente,
120     cl.APELLIDO2 AS Apellido2_Cliente,
121     c.NOMBRE AS Nombre_Comercial,
122     c.APELLIDO1 AS Apellido1_Comercial,
123     c.APELLIDO2 AS Apellido2_Comercial,
124     p.TOTAL AS Total_Venta
125 FROM UNIR_PEDIDO p
126 INNER JOIN UNIR_CLIENTE cl ON p.ID_CLIENTE = cl.ID
127 INNER JOIN UNIR_COMERCIAL c ON p.ID_COMERCIAL = c.ID;

```

Donde: **p**: Alias para tabla y campos de la tabla Pedido. **cl**: Alias para tabla y campos de la tabla Cliente. **c**: Alias para tabla y campos de la tabla Comercial. **CREATE VIEW ResumenPedidos AS**: Crea la vista de nombre ResumenPedidos. **SELECT**: Es la definición de la vista, la forma en que se obtendrán los datos cuando se consulte la vista. **FROM PEDIDO p**: Indica de qué tabla provienen los datos. **INNER JOIN CLIENTE cl ON p.ID\_CLIENTE = cl.ID**: Une los registros de la tabla PEDIDO con registros de la tabla CLIENTE donde los ID\_CLIENTE coinciden. **INNER JOIN COMERCIAL c ON p.ID\_COMERCIAL = c.ID**: Une registros de la tabla PEDIDO con registros de la tabla COMERCIAL donde los ID\_COMERCIAL coinciden. Ahora para ver el resultado de la vista creada se ejecuta la sentencia **SELECT \* FROM ResumenPedidos** y ejecuto:

	ID_Pedido	Fecha_Pedido	Nombre_Cliente	Apellido1_Cliente	Apellido2_Cliente	Nombre_Comercial	Apellido1_Comercial	Apellido2_Comercial	Total_Venta
►	1	2017-10-05	Marcos	Loyola	Méndez	Juan	Gómez	López	150.50
	2	2016-09-10	Aarón	Rivero	Gómez	Antonio	Carretero	Ortega	270.65
	3	2017-10-05	Adela	Salas	Díaz	Daniel	Sáez	Vega	65.26
	4	2016-08-17	Pepe	Ruiz	Santana	Diego	Flores	Salas	110.50
	5	2017-09-10	Marcos	Loyola	Méndez	Juan	Gómez	López	948.50
	6	2016-07-27	Pilar	Ruiz	NOVA	Daniel	Sáez	Vega	2400.60
	7	2015-09-10	Adela	Salas	Díaz	Daniel	Sáez	Vega	5760.00
	8	2017-10-10	Adrián	Suárez	NOVA	Manuel	Dominguez	Hernández	1983.43
	9	2016-10-10	Pepe	Ruiz	Santana	Diego	Flores	Salas	2480.40
	10	2015-06-27	Pepe	Ruiz	Santana	Juan	Gómez	López	250.45
	11	2016-08-17	Adolfo	Rubio	Flores	Antonio	Vega	Hernández	75.29
	12	2017-04-25	Adela	Salas	Díaz	Daniel	Sáez	Vega	3045.60
	13	2019-01-25	María	Santana	Moreno	Daniel	Sáez	Vega	545.75
	14	2017-02-02	María	Santana	Moreno	Daniel	Sáez	Vega	145.82
	15	2019-03-11	Aarón	Rivero	Gómez	Antonio	Carretero	Ortega	370.85
	16	2019-03-11	Aarón	Rivero	Gómez	Antonio	Carretero	Ortega	2389.23

## CONSULTA DE VISTA RESUMEN PEDIDOS

Utilizo las funciones agregadas como **SUM, AVG y COUNT** en una consulta con la palabra reservada **SELECT**, agrupando los resultados por el nombre y apellido del comercial y luego, se procede a ordenar por el total de ventas en orden descendente para mostrar el orden de mayor a menor. Entonces la consulta queda:

Asignatura	Datos del alumno	Fecha
Bases De Datos	Apellidos: De Mendoza	9/11/2024
	Nombre: Alejandro	

```

131 • SELECT
132     Nombre_Comercial,
133     Apellido1_Comercial,
134     Apellido2_Comercial,
135     SUM(Total_Venta) AS Total_Ventas,
136     AVG(Total_Venta) AS Promedio_Ventas,
137     COUNT(ID_Pedido) AS Cantidad_Pedidos
138 FROM
139     ResumenPedidos
140 GROUP BY
141     Nombre_Comercial,
142     Apellido1_Comercial,
143     Apellido2_Comercial
144 ORDER BY
145     Total_Ventas DESC;

```

Donde: **Select**: Elije las columnas que queremos mostrar. **SUM(Total\_Venta) como Total\_Ventas**: Se utiliza para obtener el total de ventas realizadas por cada comercial. **AVG(Total\_Venta) como Promedio\_Ventas**: Se utiliza para calcular el promedio de ventas. **COUNT(ID\_Pedido) como Cantidad\_Pedidos**: Se utiliza para contar el número de pedidos realizados por cada comercial. **FROM ResumenPedidos**: Indica que la consulta utiliza la vista ResumenPedidos como fuente de datos. **GROUP BY**: Agrupa los datos por el nombre y apellidos del comercial para que las funciones agregadas (SUM, AVG, COUNT) se calculen para cada comercial. **ORDER BY Total\_Ventas DESC**: Ordena el resultado por el total de ventas en descendente, mostrando primero el comercial con mayores ventas. Y el resultado es:

	Nombre_Comercial	Apellido1_Comercial	Apellido2_Comercial	Total_Ventas	Promedio_Ventas	Cantidad_Pedidos
►	Daniel	Sáez	Vega	11963.03	1993.838333	6
	Antonio	Carretero	Ortega	3030.73	1010.243333	3
	Diego	Flores	Salas	2590.90	1295.450000	2
	Manuel	Dominguez	Hernández	1983.43	1983.430000	1
	Juan	Gómez	López	1349.45	449.816667	3
	Antonio	Vega	Hernández	75.29	75.290000	1

El resultado nos muestra el total de ventas, el promedio de ventas y la cantidad de pedidos realizados por cada comercial, adicionalmente la consulta se presenta ordenada por el total de ventas en orden descendente. Esta correcto.

## CONCLUSIÓN

Desde mi punto de vista esta actividad ha cubierto varios aspectos clave en el diseño y manipulación de bases de datos utilizando en mi caso la plataforma MySQL, desde la creación de esquemas y tablas hasta la realización de consultas complejas y la gestión de relaciones con llaves primarias y foráneas donde se crearon tablas con relaciones (como las de cliente, pedido y comercial) permitiendo modelar información de manera estructurada y lógica. Y para mí es crucial reflejar correctamente la realidad en una base de datos. Ahora la implementación de claves primarias y foráneas asegura la integridad referencial, evitando datos duplicados con la función DISTINCT y asegurando que las relaciones entre tablas sean consistentes. Por otra parte, las restricciones ayudan a mantener la calidad y coherencia de los datos, y frente a la práctica de combinar múltiples tablas usando JOIN se demuestra la utilidad de SQL para extraer información específica de conjuntos de datos complejos. Esto es algo yo considero esencial para realizar análisis y obtener información valiosa. Ahora en mi caso durante la actividad, surgieron errores comunes, como la falta de un esquema, el modo de actualización segura o problemas de restricción de claves foráneas. Resolver estos errores para mi consideración es una parte importante del aprendizaje y de trabajar con bases de datos relacionales. En cuanto a usar alias para tablas y ordenar los resultados de las consultas me permitió además de conocer la temática, escribir consultas más legibles y organizadas, lo que mejora la mantenibilidad del código SQL y ayuda a otros a entender la lógica rápidamente. En resumen, esta actividad me ha permitido fortalecer habilidades clave para trabajar de manera eficiente en MySQL, lo que es fundamental para proyectos más grandes y complejos en mi futuro como ingeniero informático, muchas gracias.

## BIBLIOGRAFÍA

A continuación, la bibliografía implementada:

- ✓ Tema 1: Aplicaciones y propósitos de los sistemas de bases de datos. Tema 2. Bases de datos y arquitectura. Tema 3. El modelo relacional: estructura y operaciones. Tema 4. El modelo relacional: álgebra relacional extendida. Tema 5. SQL: Introducción y estructura básica. Tema 6. SQL: Operaciones y funciones. Tema 7. SQL: Consultas complejas y vistas.
- ✓ Clases virtuales con el profesor Ing. Javier Diaz Diaz.
- ✓ Material de estudio del aula.