Leonardo Sepúlveda Bedoya

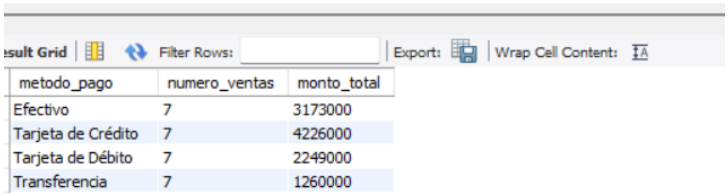
Documento con las consultas y una breve explicación de cada resultado obtenido.

1- Distribución de ventas por método de pago

- Tabla principal: `venta`
- Tablas relacionadas: `detalle_venta`
- Objetivo: Obtener el número de ventas y el monto total facturado por cada método de pago.

Para este punto , utilizamos el join que nos enseñó el profe , y este fue el resultado dado

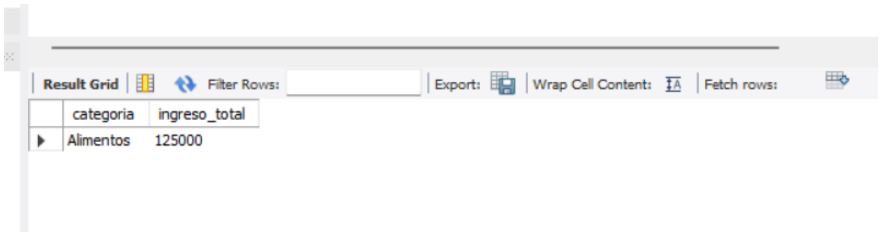
```
1
2 •  SELECT
3    v.metodo_pago,
4    COUNT(v.id) AS numero_ventas,
5    SUM(dv. cantidad* dv.precio_unitario) AS monto_total
6    FROM   tienda . venta v
7    JOIN tienda . detalle_venta dv ON v.detalle_venta_id = dv.id
8    GROUP BY v.metodo_pago;
9
```



metodo_pago	numero_ventas	monto_total
Efectivo	7	3173000
Tarjeta de Crédito	7	4226000
Tarjeta de Débito	7	2249000
Transferencia	7	1260000

2 . Categoría que genera mayor facturación

a. Tablas principales: `categoria`, `producto`, `detalle_venta`
b. Objetivo: Determinar qué categoría de producto ha generado el mayor ingreso sumando `cantidad * precio_unitario`.



categoria	ingreso_total
Alimentos	125000

En este nos solicita un join , el cual incorporamos , y este fue nuestro resultado,donde separamos , y llegamos a la categoria (alimentos) , nos arrojó su ingreso total

3 . Empleado con mejor desempeño (más ventas realizadas)

a. Tablas principales: `empleado`, `venta`
b. Objetivo: Identificar al empleado que ha registrado la mayor cantidad de ventas y mostrar cuántas ha realizado.

En este nos piden identificar al mejor empleado del mes , el cual llegamos exitosamente, donde ordenamos de manera DESC para que nos diera el id , nombre , apellidoy el total de ventas

```
SELECT
`e`.`id`,
`e`.`nombre`,
`e`.`apellido`,
COUNT(`v`.`id`) AS `total_ventas`
FROM `tienda`.`empleado` `e`
JOIN `tienda`.`venta` `v` ON `e`.`id` = `v`.`empleado_id`
GROUP BY `e`.`id`,  `e`.`nombre`, `e`.`apellido`
ORDER BY `id` DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

id	nombre	apellido	total_ventas
10	Diego	Salinas	2

4 . Evolución del inventario de un producto específico

a. Tablas principales: `producto`, `detalle_venta`, `venta`
b. Objetivo: Para un `producto_id` dado, calcular cómo ha cambiado su stock a lo largo del tiempo restando las ventas realizadas y mostrar los resultados ordenados por fecha.
Aquí , creamos una variable , como en java llamada stock , para que asi nos diera el resultado esperado

```
SET
@stock := (
SELECT `cantidad`

FROM `tienda`.`producto`
WHERE `id` = 20
);

SELECT
`v`.`createdAt`AS `fecha_venta`,
`dv`.`cantidad`AS `cantidad_vendida`,
@stock := @stock - `dv`.`cantidad`AS
`stock_restante`
FROM `tienda`.`venta` `v`
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

fecha_venta	cantidad_vendida	stock_restante
2023-05-02 14:00:00	1	44

5 . Clientes sin compras desde su alta

a. Tablas principales: `cliente`, `venta`
b. Objetivo: Listar aquellos clientes que no aparecen en ninguna venta (es decir, que aún no han comprado nada).

Aquí solicitamos para que aparezcan aquellos clientes venta no existe , entonces utilizamos el join left para que devuelva todo lo que está a la izquierda

```
1 •  SELECT `c`.*
2    FROM `cliente` `c`
3    LEFT JOIN `venta` `v` ON `c`.`id`= `v`.`cliente_id`
4    WHERE `v`.`id` IS NULL;
```


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	nombre	apellido	email	telefono	createdAt	updatedAt
1	María	Gómez	maria.gomez1@example.com	+57 312 345 6781	2023-01-10 09:15:00	NULL
4	Carlos	López	carlos.lopez4@example.com	+57 314 678 9014	2023-04-18 10:00:00	2023-10-01 12:00:00
6	Ricardo	Hernández	ricardo.hernandez6@example.com	+57 316 890 1236	2023-06-30 13:15:00	2023-09-15 09:00:00

6. Ticket promedio por venta

a. Tablas principales: `venta`, `detalle_venta`
b. Objetivo: Calcular el monto promedio de todas las ventas (sumar por venta y luego promediar).

En este punto utilizamos el AVG(promedio) , para que caclulara lo solicitado

```
1
2 •  SELECT
3    AVG(`total_venta`) AS `ticket_promedio`
4    FROM (
5    SELECT
6    `v`.`id` AS `venta_id`,
7    SUM(`dv`.`cantidad` * `dv`.`precio_unitario`) AS `total_venta`
8    FROM `tienda`.`venta` `v`
9    JOIN `detalle_venta` `dv` ON `v`.`detalle_venta_id` = `dv`.`id`
10   GROUP BY `v`.`id`) AS `venta_totales`;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ticket_promedio
389571.4285714286

7. Meses con más actualizaciones de venta

a. Tabla principal: `venta`
b. Objetivo: Identificar en qué mes(es) hubo más registros de `updatedAt` no nulo, es decir, ventas modificadas, y mostrar el conteo por mes.

Aquí utilizamos el updatedAt para que nos mostrara las fechas , y asi llegar el resultado solicitado 😊

```
SELECT
MONTH (`updatedAt`) AS `mes`,
COUNT(*) AS `modificaciones`
FROM `venta`
WHERE `updatedAt` IS NOT NULL
GROUP BY `mes`
ORDER BY `modificaciones` DESC;
```