




Taller
familiar
de

sombrillas para playa , publicidad y jardín.

- Ubicada en Ciudadela, Partido 3 de Febrero - Buenos Aires
- Ventas por mayor y menor

www.sombrillasrb.com

 Somos RB Sombrilla un taller familiar con casi 13 años de dedicación

<https://www.instagram.com/sombrillasrb/>

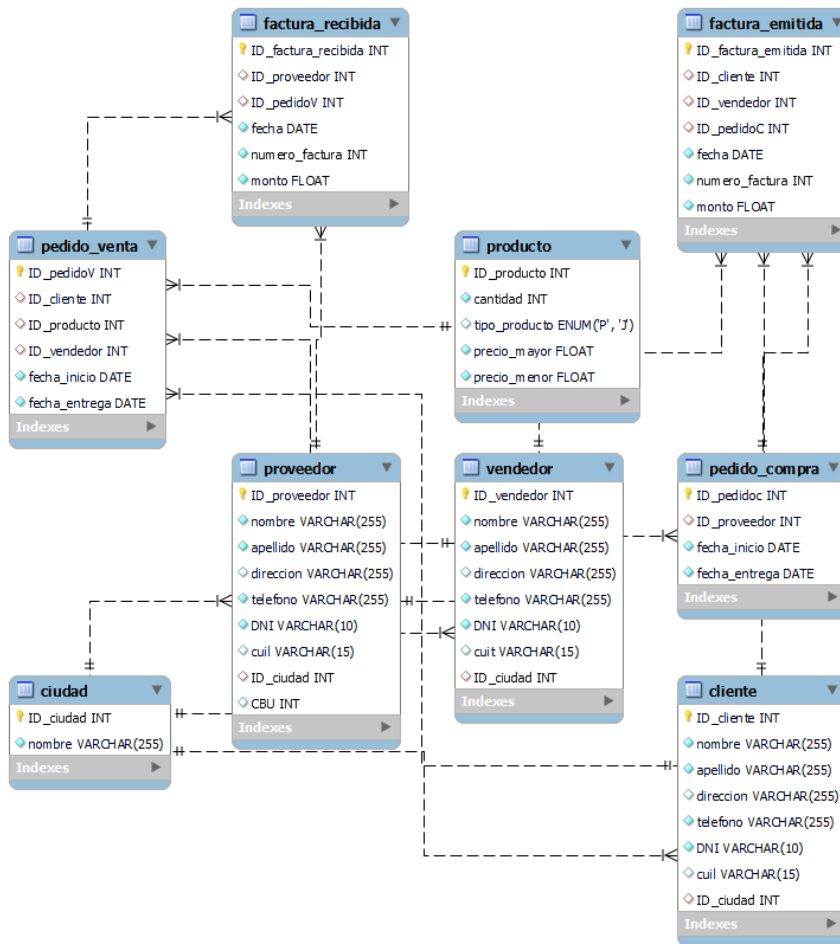
Proyecto de:

Daniela Tassara Botello

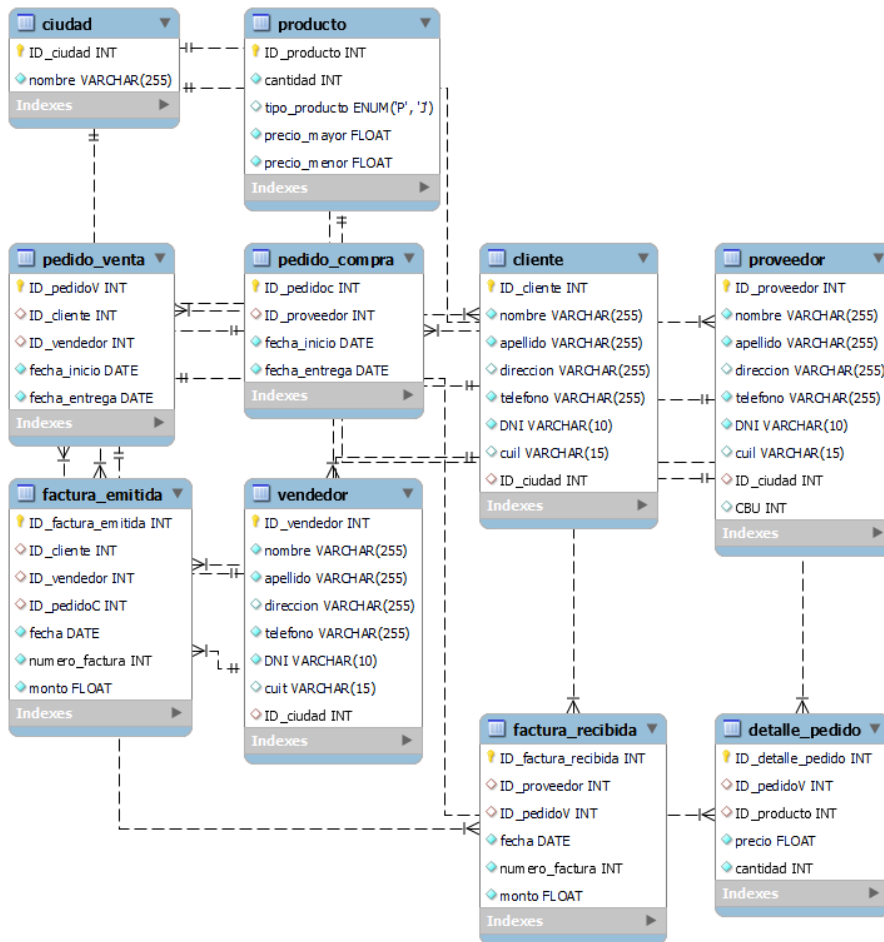
Índice :

- 1) Explicación de la empresa
- 2) Diagrama de la primer entrega
- 3) Corrección del diagrama
- 4) Creación de vistas
- 5) Creación de funciones
- 6) Creación de Stored Procedure
- 7) Creación de Triggers

Una vez creada la base de datos y las tablas , se extrae del workbench :



3) Corrección del diagrama



Generación de la tabla detalle_pedido .

4) Creación de vistas

Vista : telefono_clientes

Muestra los datos de los clientes

```

294
295  /*
296  vistas
297  */
298  • select * from cliente;
299  • CREATE OR REPLACE VIEW telefono_clientes
300  AS
301  select nombre , apellido , telefono
302  from cliente ;
303  • select *from telefono_clientes;

```

nombre	apellido	telefono
Angelico	Berzin	3537185385
Chane	Shulem	1591652117
Leesa	Hunn	5193786136
Meris	Fesby	8383764129
Corry	Finnimore	3124396095
Lacie	Serot	2022538705
Damiano	Salmoni	6078420269
Damita	Colleford	1558441274
Theda	Cutchie	9042352977
Connie	Piggre	2266220165

Vista : vista_stock_10

Muestra los productos con un stock mayor o igual a 10 unidades.

```

305
306  • CREATE OR REPLACE VIEW vista_stock_10 AS
307  SELECT ID_producto , tipo_producto , cantidad
308  FROM producto
309  WHERE ID_producto IN (SELECT ID_producto
310  FROM producto
311  WHERE cantidad >= 10);
312  • select * from vista_stock_10;

```

ID_producto	tipo_producto	cantidad
1	P	22
2	J	156
3	P	60
4	P	117
5	J	156
6	P	69
7	P	73
8	J	15
9	P	20
10	J	15

Vista: vista_compras

Muestra clientes que hayan hecho pedido durante el 2022

```

314
315 • CREATE OR REPLACE VIEW vista_compras AS
316 select c.nombre, c.apellido , c.telefono
317 from cliente as c
318 inner join pedido_venta as PV
319 on PV.ID_pedidoV = c.ID_cliente
320 where PV.fecha_entrega like '%2022%';
321
322 • select * from vista_compras;

```

nombre	apellido	telefono
Angelico	Berzin	3537185385
Chane	Shulem	1591652117
Leesa	Hunn	5193786136
Meris	Fesby	8383764129
Damiano	Salmoni	6078420269
Damita	Colleford	1558441274
Theda	Cutchie	9042352977

Vista: vista_venta_playa

Muestra los pedidos que se hicieron de productos de playa.

```

324 • create or replace view vista_ventas_playa as
325 select d.ID_producto , d.cantidad
326 from detalle_pedido as d
327 inner join producto as p
328 on p.ID_producto = d.ID_producto
329 where p.tipo_producto like '%P%';
330
331 • select * from vista_ventas_playa;

```

ID_producto	cantidad
1	58
3	67
4	190
6	175
7	177
9	131

5) Creación de funciones

Función **get_name_proveedor** recibe como parametro ID_proveedor y devuelve el nombre de ellos .

```

338
339 -- Traer el nombre de proveedores
340 • drop function get_name_proveedor ;
341
342 DELIMITER $$
343 • CREATE FUNCTION get_name_proveedor (numero_proveedor INT)
344 RETURNS VARCHAR(255)
345 READS SQL DATA
346 BEGIN
347     DECLARE resultado VARCHAR(255);
348     SET resultado = (SELECT nombre FROM proveedor WHERE ID_proveedor = numero_proveedor);
349     RETURN resultado;
350 END $$
351
352 • select get_name_proveedor(ID_proveedor) as 'Nombre Proveedor' FROM proveedor;
353

```

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

Nombre Proveedor
Delbert
De
Wheeler
Carleton
Auberta
Loni
Ara
Judith
Giralda
Reese

Función **get_según_monto** recibe como parametro el monto de las facturas en este select de las facturas emitidas , las devuelve ordenadas por fecha descendente y con la leyenda de “monto superior o menor a 60.000”


```

352 select get_name_proveedor(ID_proveedor) as 'Nombre Proveedor' FROM proveedor;
353
354 -- Traer facturas emitidas con un monton mayor a $600.000 por fecha descendente
355 drop function get_segun_monto;
356
357 DELIMITER $$
358 • CREATE FUNCTION get_segun_monto (monto_factura FLOAT)
359 RETURNS VARCHAR(255)
360 READS SQL DATA
361 BEGIN
362     CASE
363     WHEN monto_factura >= 600000 THEN RETURN 'Monto mayor a 600.000';
364     ELSE RETURN 'Monto menor';
365     END CASE;
366 END $$
367
368 • select ID_cliente , get_segun_monto(monto) as ' Montos factura', fecha, ID_vendedor from factura_emitida
369 order by fecha desc;
370

```

Result Grid

ID_cliente	Montos factura	fecha	ID_vendedor
5	Monto menor	2022-10-17	5
9	Monto mayor a 600.000	2022-10-09	9
8	Monto menor	2022-09-30	8
7	Monto mayor a 600.000	2022-07-13	7
2	Monto mayor a 600.000	2022-05-09	2
3	Monto mayor a 600.000	2022-03-29	3
4	Monto mayor a 600.000	2022-03-29	4
6	Monto mayor a 600.000	2022-02-11	6
1	Monto menor	2021-12-26	1
10	Monto mayor a 600.000	2021-12-16	10

Result 17

6) Creación de Stored Procedure

Creación de **sp_ordenar** , donde recibe el nombre de una columna de la tabla productos y los ordena de forma ascendente. Se probó con las columnas tipo_producto y cantidad .

The screenshot shows a SQL IDE with a script editor and a results grid. The script editor contains the following SQL code:

```

375  */
376  /*ORDENAR UNA TABLA MEDIANTE EL CAMPO QUE LE PASAMOS POR PARAMETROS DE MANERA ASC*/
377
378  DROP PROCEDURE sp_ordenar;
379
380  DELIMITER $$
381  CREATE PROCEDURE sp_ordenar (IN columna VARCHAR(30) )
382  BEGIN
383
384      IF columna <> '' THEN
385          SET @orden = CONCAT('ORDER BY ' , columna);
386      ELSE
387          SET @orden = '';
388      END IF;
389
390      SET @clausula = CONCAT('SELECT * FROM producto ' , @orden);
391
392      PREPARE runSQL FROM @clausula;
393      EXECUTE runSQL;
394      DEALLOCATE PREPARE runSQL;
395  END$$
396  CALL sp_ordenar ('cantidad');
397  CALL sp_ordenar ('tipo_producto');
398
399  select * from producto;
400

```

The results grid shows the output of the stored procedure for the 'cantidad' parameter. The table has 10 rows and 6 columns: ID_producto, cantidad, tipo_producto, precio_mayor, and precio_menor.

ID_producto	cantidad	tipo_producto	precio_mayor	precio_menor
1	22	P	87118	97689
3	60	P	80472	97520
4	117	P	9787	87328
6	69	P	83876	53128
7	73	P	98679	80495
9	20	P	91881	16225
2	156	J	40441	91039
5	156	J	51770	60366
8	15	J	44139	9724
10	15	J	92647	31218

Creación de **sp_mostrar_por_cantidad** , donde se ingresa un parámetro que indica la cantidad de productos dentro de un pedido y me muestra un listado donde si la cantidad que busco es menor a 100 de forma ascendente y si es mayor de forma descendente .

```

9
10 -- mostrar los pedidos que tengan cierta cantidad de productos.
11 select * from detalle_pedido;
12
13 drop procedure sp_mostrar_por_cantidad ;
14
15 DELIMITER $$
16 • CREATE PROCEDURE sp_mostrar_por_cantidad (IN p_cantidad INT)
17 BEGIN
18     IF p_cantidad >= 100 THEN
19         SELECT * FROM detalle_pedido WHERE cantidad >= p_cantidad ORDER BY cantidad DESC;
20     ELSE
21         SELECT * FROM detalle_pedido WHERE cantidad <= p_cantidad ORDER BY cantidad ASC;
22     END IF;
23 END$$
24
25 • call sp_mostrar_por_cantidad (110);
26 call sp_mostrar_por_cantidad (80);

```

ID_detalle_pedido	ID_pedidoV	ID_producto	precio	cantidad
5	5	5	530043	9
1	1	1	995296	58
3	3	3	481516	67
2	2	2	811865	72

7) Creación de triggers

Creación del primer trigger **producto_historico** para un histórico, donde se guarden los productos nuevos cargados en la tabla “producto” .

```
clase10  rb_Sombrillas* x
Limit to 1000 rows

426 CREATE TABLE IF NOT EXISTS producto_historico (
427     ID_p INT ,
428     tipo_p enum('P' , 'J'), -- de playa o jardin
429     precio_may FLOAT,
430     precio_men FLOAT,
431     fecha_hora datetime
432 );
433
434 DELIMITER //
435 • CREATE TRIGGER trigger_producto_historico
436 AFTER INSERT ON producto -- after porque es despues de ingresado
437 FOR EACH ROW -- por cada uno de los registros
438 BEGIN
439     INSERT INTO producto_historico (ID_p,tipo_p,precio_may, precio_men,fecha_hora)
440     VALUES (NEW.ID_producto, NEW.tipo_producto,new.precio_mayor,new.precio_menor, NOW());
441 END//
442
443 • drop trigger trigger_producto_historico;
444
445 select * from producto_historico;
446
447 INSERT INTO producto VALUES (11,10,'P', 11000, 23000);
```

<

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









	ID_p	tipo_p	precio_may	precio_men	fecha_hora
▶	11	P	11000	23000	2022-12-13 23:05:43

Creación del primer trigger **cliente_historico** para un histórico, donde se guarden los clientes nuevos cargados en la tabla “cliente” .






Donde es importante registrar también la fecha.

0

rb_Sombrillas*



Limit to 1000 rows



-- trigger que guarde los clientes nuevos
drop table cliente_historico;



CREATE TABLE IF NOT EXISTS cliente_historico (
 ID_client INT ,
 nombre varchar(255) not null,
 apellido varchar(255) not null,
 telefono VARCHAR(255) not null,
 fecha_hora datetime
);

DELIMITER //
• CREATE TRIGGER trigger_cliente_historico
 AFTER INSERT ON cliente
 FOR EACH ROW
 BEGIN
 INSERT INTO cliente_historico (ID_client,nombre,apellido, telefono,fecha_hora)
 VALUES (new.ID_cliente, new.nombre, new.apellido, new.telefono, NOW());
 END//

• INSERT INTO cliente VALUES (null,'Daniela','Tassara','cabaliito','3537185385','167722405','493829149',5);


select * from cliente_historico;

t Grid




Filter Rows:

Export:



Wrap Cell Content:



ID_client

nombre

apellido

telefono

fecha_hora

Daniela

Tassara

3537185385

2022-12-13 23:27:59