

Nombre de la práctica	Vistas en Navicat con Mysql			No.	
Asignatura:	Taller de Base de Datos	Carrera:	Ing. Sistemas Computacionales	Duración de la práctica (Hrs)	

Raúl Ciriaco Castillo 3501 ISIC

I. Competencia(s) específica(s):

II. Lugar de realización de la práctica (laboratorio, taller, aula u otro): Aula

III. Material empleado:

- Computadora
- Software Navicat proporcionado por el docente

IV. Desarrollo de la práctica:

1. Creación de base de datos:

Primero que nada, comenzaremos a crear nuestra base de datos la cual nombraremos como “puntos” en nuestro espacio de trabajo dentro de Navicat.

2. Creación de Tablas:

Pasaremos a definir y crear nuestras tablas a utilizar dentro de nuestra base de datos, en este caso como se muestra en la siguiente imagen:

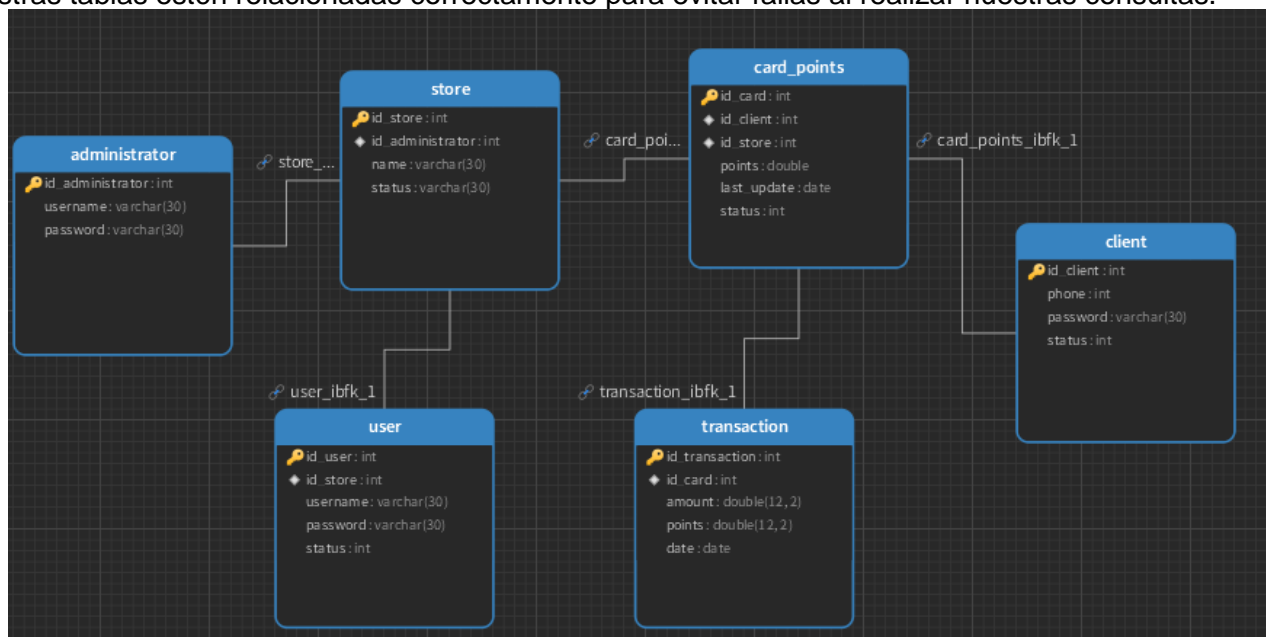
```

1
2  # CREACION DE TABLAS
3  CREATE TABLE administrator(
4      id_administrator INT PRIMARY KEY AUTO_INCREMENT,
5      username VARCHAR(30) NOT NULL,
6      password VARCHAR(30) NOT NULL
7  );
8  CREATE TABLE store(
9      id_store INT PRIMARY KEY AUTO_INCREMENT,
10     id_administrator INT,
11     name varchar(30) NOT NULL,
12     status varchar(30) NOT NULL,
13     FOREIGN KEY(id_administrator) REFERENCES administrator(id_administrator) ON UPDATE CASCADE ON DELETE RESTRICT
14 );
15 CREATE TABLE user(
16     id_user INT PRIMARY KEY AUTO_INCREMENT,
17     id_store INT,
18     username VARCHAR(30) NOT NULL,
19     password VARCHAR(30) NOT NULL,
20     status int NOT NULL,
21     FOREIGN KEY(id_store) REFERENCES store(id_store) ON UPDATE CASCADE ON DELETE RESTRICT
22 );
23 CREATE TABLE client(
24     id_client INT PRIMARY KEY AUTO_INCREMENT,
25     phone int NOT NULL,
26     password VARCHAR(30) NOT NULL,
27     status int NOT NULL
28 );
29 CREATE TABLE card_points(
30     id_card INT PRIMARY KEY AUTO_INCREMENT,
31     id_client INT,
32     id_store INT,
33     points DOUBLE NOT NULL,
34     last_update DATE NOT NULL,
35     status int NOT NULL,
36     FOREIGN KEY(id_client) REFERENCES client(id_client) ON UPDATE CASCADE ON DELETE RESTRICT,
37     FOREIGN KEY(id_store) REFERENCES store(id_store) ON UPDATE CASCADE ON DELETE RESTRICT
38 );
39 CREATE TABLE transaction(
40     id_transaction INT PRIMARY KEY AUTO_INCREMENT,
41     id_card INT,
42     amount DOUBLE(12,2) NOT NULL,
43     points DOUBLE(12,2) NOT NULL,
44     date DATE NOT NULL,
45     FOREIGN KEY(id_card) REFERENCES card_points(id_card) ON UPDATE CASCADE ON DELETE RESTRICT
46 );
47

```

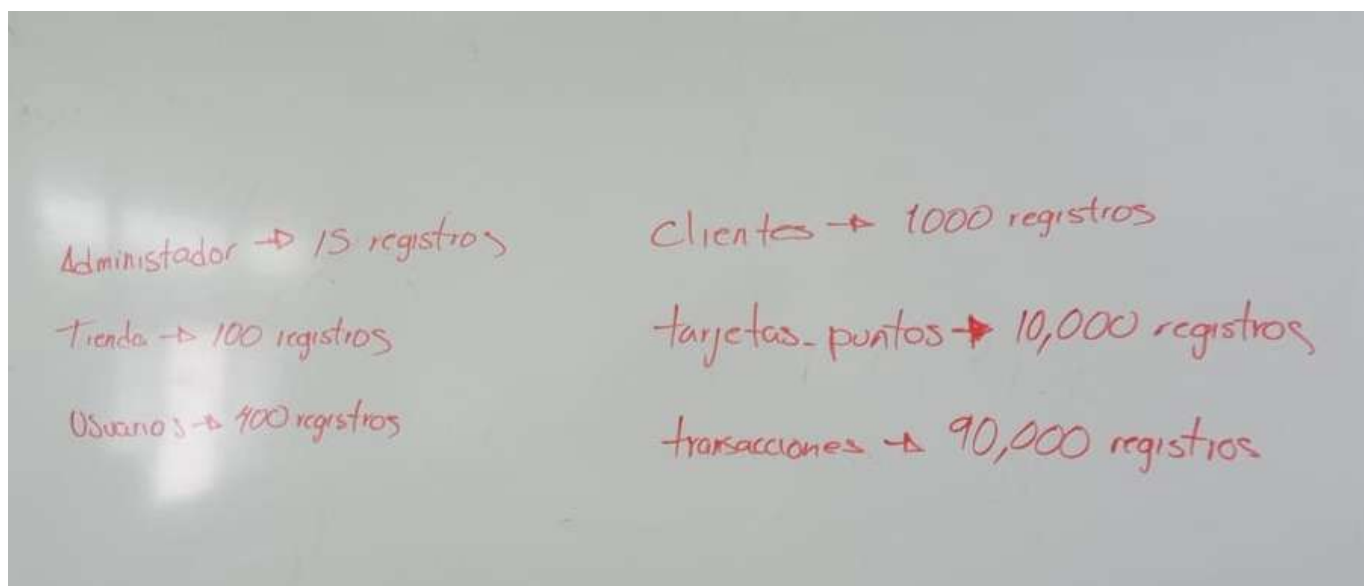
3. Diagramas de Clase Generadas

Una vez definidas nuestras tablas procedemos a analizar el diagrama de las mismas, asegurando que nuestras tablas estén relacionadas correctamente para evitar fallas al realizar nuestras consultas.



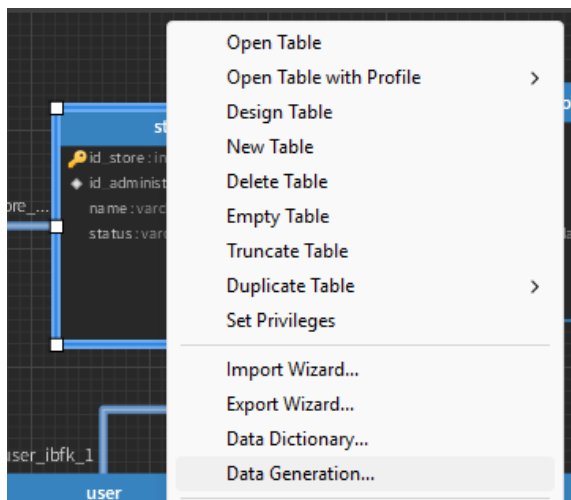
4. Ingesta de datos para cada Tabla existente:

De acuerdo a las especificaciones dadas por el docente para cada tabla en cuanto a los registros que deben de existir, tomaremos las siguientes:

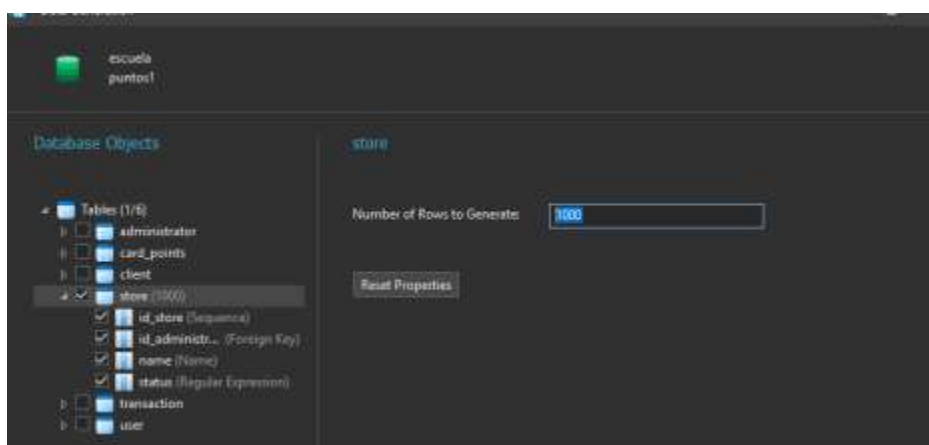


Como haremos esto, primero que nada, seleccionaremos la tabla a la cual vayamos a generarle los datos como se muestra a continuación:

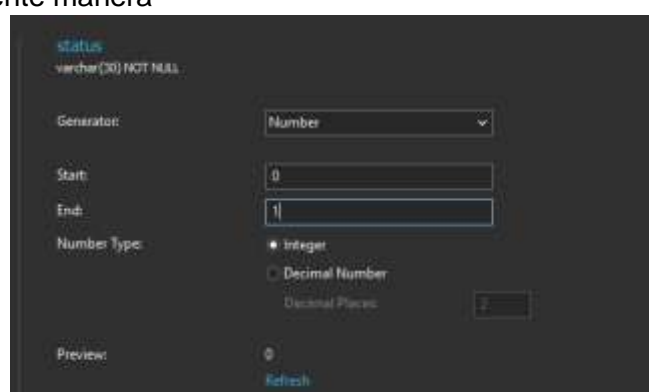
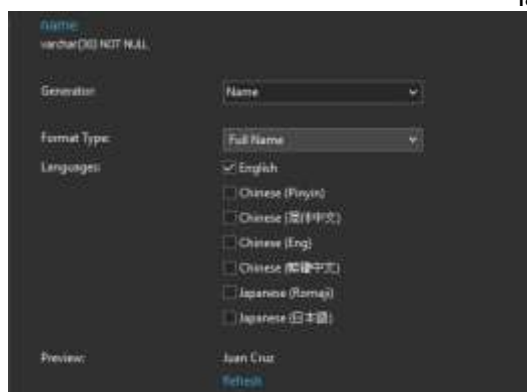
Vamos a seleccionar la tabla dando click derecho y seleccionaremos el apartado que dice “Data Generation”.



Aquí podremos colocar le número de datos que deseamos insertar para nuestras tablas:



Pulsaremos en los atributos para especificar y editarlos tanto para Name, status como se muestra de la siguiente manera



Pulsaremos next y nos mostrara la manera en que se insertaran los datos, permitiendo generar las veces que sea necesarias los datos:

Preview

Table: store Regenerate

id_store	id_administrator	name	status
201	4	Todd Smith	HqQhKRRH
202	10	Sato Yamato	IAQK1vWtpe
203	8	Feng Ziyi	KZGZwhuA
204	11	Norman Schmidt	XzBX7Mvbk
205	12	Taniguchi Sara	3TdcST7vz
206	9	Karada Ayano	uZGhwhHd
207	7	Ariya Ford	RhoDCAgeI
208	12	Alia Ayano	IBdy7at21
209	2	Aeng Lu	vSQMLmAX
210	6	Yamamoto Haru	KvYHhAqdv
211	7	Tian Anqi	fpRkZfE8
212	12	Yau Hui Tung	H8RDUvWtW
213	3	Yamashita Takuya	uRfGdyp1
214	14	Benjamin Ellis	HRYM41ZEV
215	7	Mak Hui Yau	ZEGKqpgy
216	14	Betty Freeman	PvHfYcwv8
217	10	Sasaki Mio	rRmCgCf6

Save Profile Load Profile Options Back Start

Pulsamos en start y nos comenzara a realizar el proceso de insertar los datos tomando en cuentas las especificaciones dadas anteriormente en los atributos, una vez terminado nos debería mostrar los datos de la siguiente manera:

100% - Data Generation

Objects:	1
Generated Records:	1,000
Inserted Records:	1,000
Errors:	0
Time:	00:00.05

[DGR] Data Generation started
 [DGR] 1> transaction: --Transaction - Begin--
 [DGR] 1> transaction: Preparing field
 [DGR] 1> transaction: Start records generation
 [DGR] 1> transaction: Records generated
 [DGR] 1> transaction: --Transaction - Commit--
 [DGR] Finished successfully

Comprobamos con un SELECT * FROM store; para rectificar que correctamente se hayan insertado nuestros datos en cada una de las tablas.

escuela puntos1 Run Selected Explain Selected

```

40 id_transaction INT PRIMARY KEY AUTO_INCREMENT,
41 id_card INT,
42 amount DOUBLE(12,2) NOT NULL,
43 points DOUBLE(12,2) NOT NULL,
44 date DATE NOT NULL,
45 FOREIGN KEY(id_card) REFERENCES card_points(id_card) ON UPDATE CASCADE ON DELETE RESTRICT
46 );
47
48 SELECT * FROM administrator;
49 SELECT * FROM store;
50 SELECT * FROM client;
51 SELECT * FROM transaction;
52 SELECT * FROM card_points;
  
```

Message: Summary Result 1

id_store	id_administrator	name	status
1	12	Bryan Peterson	1
2	2	Larry Miller	0
3	3	Andrew Rose	1
4	8	Edna Chavez	0
5	14	Jonathan Richards	1
6	6	Rita Mitchell	1
7	9	Anthony Garcia	0
8	5	Lois Selman	1
9	4	Susan Gutierrez	1

Una vez realizado la inserción de cada una de las tablas podemos continuar con la siguiente fase:

5. Creación de Vistas

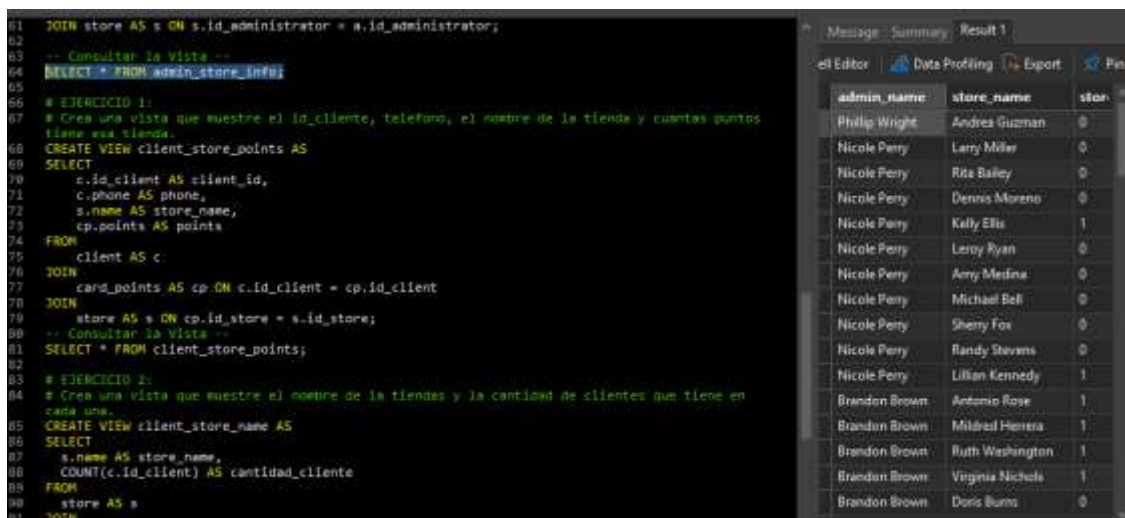
Este ejemplo que se muestra a continuación nos crea una vista que combina información de las tablas **administrador y store**, proporcionando una manera simplificada de acceder a los nombres de usuario de los administradores junto con los nombres y estados de las tiendas que administran.

Y después realizamos la consulta que nos mostrara el contenido de nuestra vista con el SELECT en la parte inferior.

```
-- Crear Vistas --
CREATE VIEW admin_store_info AS
SELECT a.username AS admin_name,
s.name AS store_name,
s.status AS store_status
FROM administrador AS a
JOIN store AS s ON s.id_administrator = a.id_administrator;

-- Consultar la Vista --
SELECT * FROM admin_store_info;
```

Muestra de pantalla según la consulta:



admin_name	store_name	store_status
Philip Wright	Andrea Guzman	0
Nicole Perry	Larry Miller	0
Nicole Perry	Rita Bailey	0
Nicole Perry	Dennis Moreno	0
Nicole Perry	Kelly Ellis	1
Nicole Perry	Leroy Ryan	0
Nicole Perry	Amy Medina	0
Nicole Perry	Michael Bell	0
Nicole Perry	Sherry Fox	0
Nicole Perry	Randy Stevens	0
Nicole Perry	Lillian Kennedy	1
Brandon Brown	Antonio Rose	1
Brandon Brown	Mildred Herrera	1
Brandon Brown	Ruth Washington	1
Brandon Brown	Virginia Nichols	1
Brandon Brown	Doris Burns	0

6. Ejercicios Resueltos en Clase

```
# EJERCICIO 1:
# Crea una vista que muestre el id_cliente, telefono, el nombre de la tienda y cuantas puntos tiene esa tienda.
CREATE VIEW client_store_points AS
SELECT
c.id_client AS client_id,
c.phone AS phone,
s.name AS store_name,
cp.points AS points
FROM
client AS c
JOIN
card_points AS cp ON c.id_client = cp.id_client
JOIN
store AS s ON cp.id_store = s.id_store;
-- Consultar la Vista --
SELECT * FROM client_store_points;
```

Muestra de pantalla según la consulta:

```

80 -- Consultar la Vista --
81 SELECT * FROM client_store_name;
82
83 # EJERCICIO 2:
84 # Crea una vista que muestre el nombre de la tiendas y la cantidad de clientes que tiene en cada una.
85 CREATE VIEW client_store_name AS
86 SELECT
87     s.name AS store_name,
88     COUNT(c.id_client) AS cantidad_cliente
89 FROM
90     store AS s
91 JOIN
92     card_points AS cp ON cp.id_store = s.id_store
93 JOIN
94     client AS c ON c.id_client = cp.id_client
95 GROUP BY
96     s.name;
97
98 -- Consultar la Vista --
99 SELECT * FROM client_store_name;
100
101 # EJERCICIO TABLA:
102 # Crea una vista que muestre el nombre de cada tienda y el monto total generado en ella.
103 CREATE VIEW store_monto AS
104 SELECT
105     s.name AS store_name,
106     -- con la operación SUM vamos a mostrar el monto total generado de cada tienda --
107     SUM(t.amount) AS monto_total
108 FROM
109     store AS s
110 JOIN
111     card_points AS tp ON tp.id_store = s.id_store
112 JOIN
113     transaction AS t ON t.id_card = tp.id_card
114 -- agrupa los campos para que no estén de manera repetida en una sola --
115 GROUP BY
116     s.name;
117
118 -- Consultar la Vista --
119 SELECT * FROM store_monto;
120
121
122
123
124
125
126
127
128
129
130
131

```

client_id	phone	store_name	points
888	552	Bryan Peterson	274.21
854	960	Bryan Peterson	620.29
486	978	Bryan Peterson	492.25
878	465	Bryan Peterson	864.85
254	128	Bryan Peterson	690.32
715	220	Bryan Peterson	87.4
825	988	Bryan Peterson	108.44
666	236	Bryan Peterson	452.88
910	188	Bryan Peterson	81.88
252	79	Bryan Peterson	12.75
705	332	Bryan Peterson	128.73
287	737	Bryan Peterson	812.88
892	247	Bryan Peterson	219.7
48	728	Bryan Peterson	288.57
235	593	Bryan Peterson	888.88
343	823	Bryan Peterson	134.4
825	983	Bryan Peterson	768.61
79	218	Bryan Peterson	104.84
527	849	Bryan Peterson	84.6
84	907	Bryan Peterson	537.52
798	794	Bryan Peterson	817.61
777	787	Bryan Peterson	864.33
343	607	Bryan Peterson	888.81
342	110	Bryan Peterson	380.8

Ejercicio #2:

```

# EJERCICIO 2:
# Crea una vista que muestre el nombre de la tiendas y la cantidad de clientes que tiene en cada una.
CREATE VIEW client_store_name AS
SELECT
    s.name AS store_name,
    COUNT(c.id_client) AS cantidad_cliente
FROM
    store AS s
JOIN
    card_points AS cp ON cp.id_store = s.id_store
JOIN
    client AS c ON c.id_client = cp.id_client
GROUP BY
    s.name;

```

Muestra de pantalla:

```

87 -- Consultar la Vista --
88 SELECT * FROM client_store_name;
89
90
91 # EJERCICIO TABLA:
92 # Crea una vista que muestre el nombre de cada tienda y el monto total generado en ella.
93 CREATE VIEW store_monto AS
94 SELECT
95     s.name AS store_name,
96     -- con la operación SUM vamos a mostrar el monto total generado de cada tienda --
97     SUM(t.amount) AS monto_total
98 FROM
99     store AS s
100 JOIN
101     card_points AS tp ON tp.id_store = s.id_store
102 JOIN
103     transaction AS t ON t.id_card = tp.id_card
104 -- agrupa los campos para que no estén de manera repetida en una sola --
105 GROUP BY
106     s.name;
107
108 -- Consultar la Vista --
109 SELECT * FROM store_monto;
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131

```

store_name	cantidad_cliente
Bryan Peterson	87
Larry Miller	102
Antonio Rose	107
Edna Chavez	98
Jonathan Richardson	107
Rita Mitchell	96
Anthony Garcia	95
Lon Salazar	104
Susan Gutierrez	95
Rita Bailey	89
Dennis Moreno	99
Rhonda Johnson	79


```
# EJERCICIO TAREA:
# Crea una vista que muestre el nombre de cada tienda y el monto total generados en ella.
CREATE VIEW store_monto AS
SELECT
    s.name AS store_name,
    -- con la operación SUM vamos a mostrar el monto total generado de cada tienda --
    SUM(t.amount) AS monto_total
FROM
    store AS s
JOIN
    card_points AS tp ON tp.id_store = s.id_store
JOIN
    transaction AS t ON t.id_card = tp.id_card
-- agrupa los campos para que no esten de manera repetida en uno solo --
GROUP BY
    s.name;
```

store_monto @puntos1 (escuela) - View	
store_name	monto_total
Bryan Peterson	421344.06
Larry Miller	447142.22
Antonio Rose	482903.65
Edna Chavez	410942.13
Jonathan Richardson	470184.96
Rita Mitchell	399081.00
Anthony Garcia	424801.07
Lois Salazar	482509.75
Susan Gutierrez	434675.09
Rita Bailey	397360.68
Dennis Moreno	435889.40
Rhonda Johnson	308845.91
Bonnie Owens	454782.80
Patricia Wells	415379.87
Jack Brown	497909.37
Clara Moore	454552.29
Roger Mendez	468782.12
Bradley Scott	378024.24
Mildred Herrera	457190.67
Johnny Mills	387751.15
Kelly Ellis	422733.51

7. Código Final elaborado en clase:

```
CREATE TABLE administrator(  
  id_administrator INT PRIMARY KEY AUTO_INCREMENT,  
  username VARCHAR(30) NOT NULL,  
  password VARCHAR(30) NOT NULL  
);  
CREATE TABLE store(  
  id_store INT PRIMARY KEY AUTO_INCREMENT,  
  id_administrator INT,  
  name varchar(30) NOT NULL,  
  status varchar(30) NOT NULL,  
  FOREIGN KEY(id_administrator) REFERENCES administrator(id_administrator) ON UPDATE  
CASCADE ON DELETE RESTRICT  
);  
CREATE TABLE user(  
  id_user INT PRIMARY KEY AUTO_INCREMENT,  
  id_store INT,  
  username VARCHAR(30) NOT NULL,  
  password VARCHAR(30) NOT NULL,  
  status int NOT NULL,  
  FOREIGN KEY(id_store) REFERENCES store(id_store) ON UPDATE CASCADE ON DELETE  
RESTRICT  
);  
CREATE TABLE client(  
  id_client INT PRIMARY KEY AUTO_INCREMENT,  
  phone int NOT NULL,  
  password VARCHAR(30) NOT NULL,  
  status int NOT NULL  
);  
CREATE TABLE card_points(  
  id_card INT PRIMARY KEY AUTO_INCREMENT,  
  id_client INT,  
  id_store INT,  
  points DOUBLE NOT NULL,  
  last_update DATE NOT NULL,  
  status int NOT NULL,  
  FOREIGN KEY(id_client) REFERENCES client(id_client) ON UPDATE CASCADE ON DELETE  
RESTRICT,  
  FOREIGN KEY(id_store) REFERENCES store(id_store) ON UPDATE CASCADE ON DELETE  
RESTRICT  
);  
CREATE TABLE transaction(  
  id_transaction INT PRIMARY KEY AUTO_INCREMENT,  
  id_card INT,  
  amount DOUBLE(12,2) NOT NULL,  
  points DOUBLE(12,2) NOT NULL,  
  date DATE NOT NULL,  
  FOREIGN KEY(id_card) REFERENCES card_points(id_card) ON UPDATE CASCADE ON  
DELETE RESTRICT  
);
```



```
SELECT * FROM administrator;  
SELECT * FROM store;  
SELECT * FROM client;  
SELECT * FROM transaction;  
SELECT * FROM card_points;  
SELECT * FROM user;
```

-- Crear Vistas --

```
CREATE VIEW admin_store_info AS  
SELECT a.username AS admin_name,  
s.name AS store_name,  
s.status AS store_status  
FROM administrator AS a  
JOIN store AS s ON s.id_administrator = a.id_administrator;  
-- Consultar la Vista --  
SELECT * FROM admin_store_info;
```

EJERCICIO 1:

Crea una vista que muestre el id_cliente, telefono, el nombre de la tienda y cuantas puntos tiene esa tienda.

```
CREATE VIEW client_store_points AS  
SELECT  
c.id_client AS client_id,  
c.phone AS phone,  
s.name AS store_name,  
cp.points AS points  
FROM  
client AS c  
JOIN  
card_points AS cp ON c.id_client = cp.id_client  
JOIN  
store AS s ON cp.id_store = s.id_store;  
-- Consultar la Vista --  
SELECT * FROM client_store_points;
```

EJERCICIO 2:

Crea una vista que muestre el nombre de la tiendas y la cantidad de clientes que tiene en cada una.

```
CREATE VIEW client_store_name AS  
SELECT  
s.name AS store_name,  
COUNT(c.id_client) AS cantidad_cliente  
FROM  
store AS s  
JOIN  
card_points AS cp ON cp.id_store = s.id_store  
JOIN  
client AS c ON c.id_client = cp.id_client  
GROUP BY
```

```
s.name;
-- Consultar la Vista --
SELECT * FROM client_store_name;

# EJERCICIO TAREA:
# Crea una vista que muestre el nombre de cada tienda y el monto total generados en ella.
CREATE VIEW store_monto AS
SELECT
    s.name AS store_name,
    -- con la operación SUM vamos a mostrar el monto total generado de cada tienda --
    SUM(t.amount) AS monto_total
FROM
    store AS s
JOIN
    card_points AS tp ON tp.id_store = s.id_store
JOIN
    transaction AS t ON t.id_card = tp.id_card
-- agrupa los campos para que no esten de manera repetida en uno solo --
GROUP BY
    s.name;
-- Consultar la Vista --
SELECT * FROM store_monto;
```

V. Conclusiones:

Este tema en Mysql sobre vistas la verdad que al principio se me hace un poco complicado debido a que no entiendo realmente los pasos para los cuales debemos de hacer el proceso de combinar las tablas entiendo las variables que usamos pero me cuesta entender la Unión de cada una de ellas y los diferentes tipos que existen al igual que definir las variables que vamos a utilizar con otras nuevas como lo hicimos anteriormente, pero con ayuda de algunos compañeros y profesor en clase pude resolver mis dudas logrando realizar mi trabajo y mi tarea correctamente como lo solicita el docente en cada ejemplo y oración de los ejercicios mostrados anteriormente.

Y la verdad esperamos seguir trabajando de esta manera realizando prácticas que nos permitan agilizar nuestro conocimiento en este ámbito de nuestra carrera para poder desarrollar de una mejor manera las bases de datos en un futuro y solucionar problemas de la vida real más adelante.