

SEGUNDA PRE-ENTREGA TP FINAL

TEMATICA: MODELO DE RENTABILIDAD DE UNA ENTIDAD FINANCIERA

Alumno: Ma. Emilia Peña Onganía

Indice

1.	Introducción	Pág. 02
2.	Objetivo - Situación Problemática y Modelo de Negocio	Pág. 02
3.	Diagrama de entidad relación	Pág. 03
4.	Listado de Tablas	Pág. 04
5.	Creación de Tablas	Pág. 07
6.	Inserción de Datos	Pág. 09
7.	Store Procedure	Pág. 11
8.	Views	Pág. 13
9.	Funciones	Pág. 16
10.	Trigger	Pág. 18
11.	Herramientas y tecnologías utilizadas	Pág. 20
12.	Líneas Futuras	Pág. 20
13.	Enlace a repositorio Github	Pág. 20

Temática elegida: Modelo de rentabilidad

Introducción:

El presente trabajo se desarrolló en base a un modelo acotado de rentabilidad de una entidad financiera.

Dada la naturaleza de este tipo de negocios los modelos de rentabilidad son complejos ya que no solo poseen productos activos sino también pasivos , además de que poseen un gran número de costos transaccionales y un mayor volumen de información para procesar.

Objetivo del Modelo:

El modelo de rentabilidad es de vital importancia en una entidad ya que permite identificar tanto aquellos clientes que son más rentables como los productos, además, si se desea, en caso de poseer varias sucursales también se podría evaluar estas.

Este modelo puede ser utilizado tanto por:

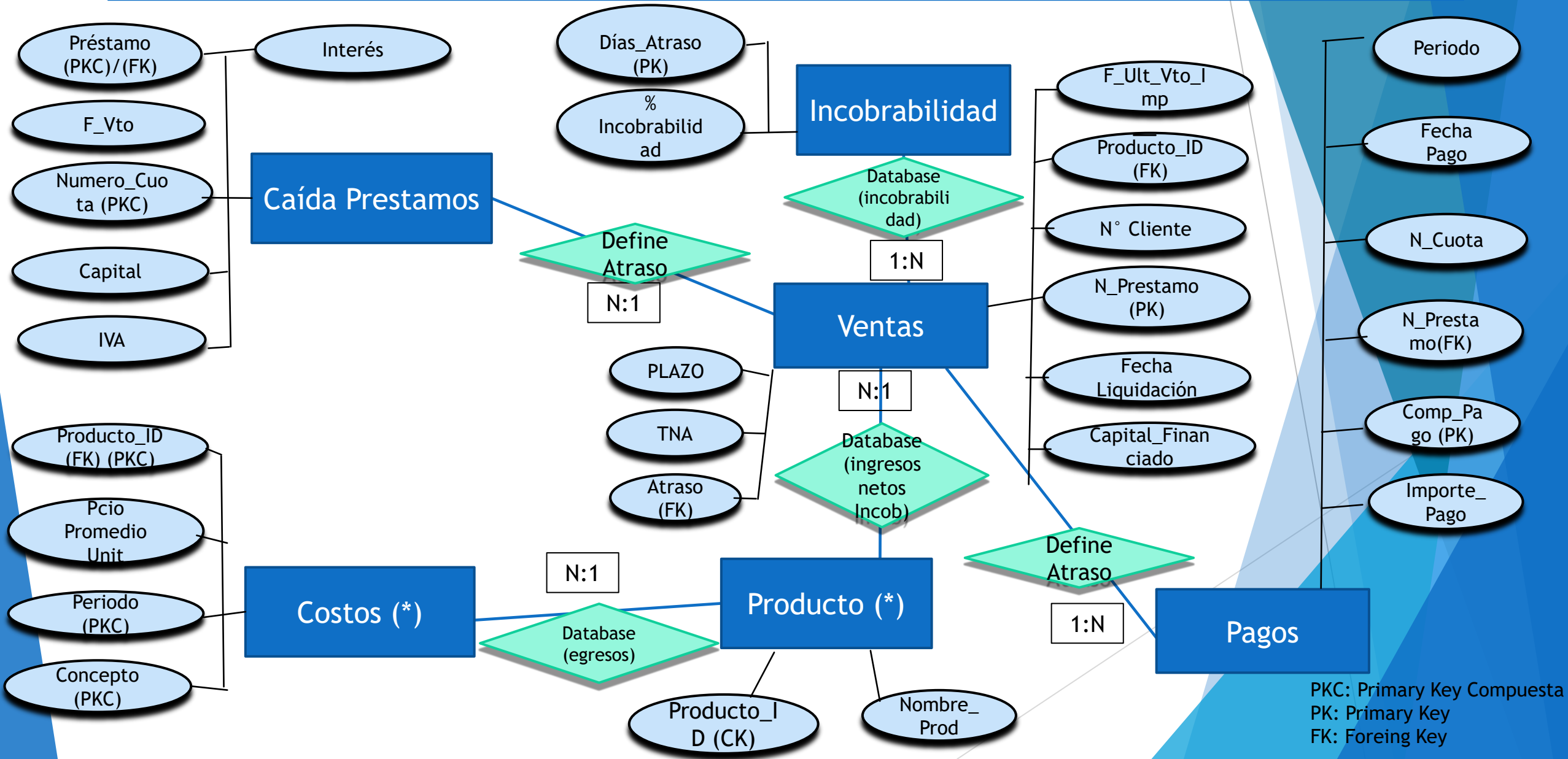
1. Áreas comerciales para:
 - ▶ elaboración de campañas, ofreciendo mejores ofertas a clientes más rentables
 - ▶ Evaluación de los productos que lideran para implementar mejoras en estos, midiendo los costos vs ingresos
 - ▶ Entre otros
2. Área de Control de Gestión:
 - ▶ Detectar ineficiencias en el desarrollo del negocio
 - ▶ Proponer mejoras
 - ▶ Medir la rentabilidad de la entidad
 - ▶ Elaborar indicadores, entre otras cosas
3. Directorio:
 - ▶ Toma de decisiones tales como: discontinuar productos poco rentables, premiar desempeños comerciales, en el caso de poseer muchas sucursales, determinar el cierre de alguna sucursal poco rentable.

Este modelo incluirá información tal como:

- ▶ información de ventas,
- ▶ información costos,
- ▶ información de pagos
- ▶ Tablas que hacen a la identificación de la mora del cliente, así como también al producto.

En este trabajo, para simplificar supondremos que se trata de una entidad con una única sucursal y mediremos los distintos productos de prestamos

Modelos relacionales



Descripción de tablas

TABLAS: MODELO DE RENTABILIDAD

TIPO DE EMPRESA: ENTIDAD FINANCIERA

NOMBRE TABLA	PRODUCTO		
DESCRIPCION	Tabla con descripción del nombre del producto		
NOMBRE CAMPO	NOMBRE CAMPO ABREVIADO	CLAVES	TIPO DE DATO
Producto_ID	Producto_ID	PK	int
Nombre_Producto	Nombre_Producto		Varchar(20)

NOMBRE TABLA	COSTOS		
DESCRIPCION	Costo promedio de cada componente de un producto		
NOMBRE CAMPO	NOMBRE CAMPO ABREVIADO	CLAVES	TIPO DE DATO
Periodo	Periodo	PKC	int
Producto_ID	Producto_ID	PKC/FK	int
Concepto	Concepto	PKC	varchar(20)
Precio_Promedio_Unitario	Pcio_Prom_Unit		float

NOMBRE TABLA	INCOBRABILIDAD		
DESCRIPCION	% de deuda que se considera incobrable según el atraso del cliente		
NOMBRE CAMPO	NOMBRE CAMPO ABREVIADO	CLAVES	TIPO DE DATO
Días_Atraso	Atraso	PK	int
Porcentaje_Incobrabilidad	%_Incob		float

Descripción de tablas (Cont.)

DESCRIPCION	Detalle de préstamos dados de alta		
NOMBRE CAMPO	NOMBRE CAMPO ABREVIADO	CLAVES	TIPO DE DATO
Numero_Cliente	N_Cliente		int
Numero_Prestamo	N_Prestamo	PK	int
Fecha_Liquidacion	Fecha_Liquidacion		date
Producto_ID	Producto_ID	FK	int
Plazo	Plazo		int
TNA	TNA		float
Capital_Financiado	Capital_Financiado		float
Atraso	Atraso	FK	int
Fecha_Ultimo_Vencimiento_Impago	F_Ult_Vto_Imp		date

NOMBRE TABLA	CAIDA PRESTAMO		
DESCRIPCION	Detalle de cuota de Préstamos		
NOMBRE CAMPO	NOMBRE CAMPO ABREVIADO	CLAVES	TIPO DE DATO
Numero_Prestamo	N_Prestamo	PKC / FK	int
Fecha_Vencimiento_Cuota	F_Vto		date
Numero_Cuota	N_Cuota	PKC	int
Capital	Capital		float
Interes	Interes		float
IVA	IVA		float

Descripción de tablas (Cont.)

NOMBRE TABLA	PAGOS		
DESCRIPCION	Detalle de pagos de clientes		
NOMBRE CAMPO	NOMBRE CAMPO ABREVIADO	CLAVES	TIPO DE DATO
Periodo	Periodo		int
Comprobante_Pago	Comp_Pago	PK	int
Numero_Prestamo	N_Prestamo	FK	int
Numero_Cuota	N_Cuota		int
Fecha_Pago	F_Pago		date
Importe_Pago	Importe_Pago		float

PK:Primary Key

FK:Foreign Key

PKC:Primary Key Compuesta

Creación de tablas

► CREA TABLA PRODUCTO:

```
create table producto (PRODUCTO_ID int auto_increment primary key, NOMBRE_PRODUCTO varchar (20));
```

► CREA TABLA COSTOS:

```
create table costos (PRODUCTO_ID int, PERIODO int, CONCEPTO varchar(20), primary key (CONCEPTO, PERIODO, PRODUCTO_ID), PCIO_PROM_UNIT float, foreign key (PRODUCTO_ID) references producto(PRODUCTO_ID));
```

CREA TABLA INCOBRABILIDAD:

```
create table incobrabilidad (ATRASO int primary key, PORCENT_INCOB float);
```

► CREA TABLA VENTAS:

```
create table ventas (N_CLIENTE int, N_PRESTAMO int auto_increment primary key, FECHA_LIQUIDACION date, PRODUCTO_ID int, PLAZO int, TNA float, CAP_FINANCIADO float, ATRASO int, F_ULT_VTO_IMPAGO date, foreign key(ATRASO) references incobrabilidad(ATRASO), foreign key (PRODUCTO_ID) references producto(PRODUCTO_ID));
```

► CREA TABLA CAÍDA PRESTAMOS:

```
create table caida_prestamos (N_PRESTAMO int,N_CUOTA int,primary key(N_PRESTAMO,N_CUOTA), F_VTO date, CAPITAL float, INTERES float, IVA float, foreign key(N_PRESTAMO) references ventas(N_PRESTAMO));
```

► CREA TABLA PAGOS:

```
create table pagos (PERIODO int,COMP_PAGO int auto_increment primary key, N_PRESTAMO int, N_CUOTA int, F_PAGO date, IMPORTE int,foreign key(N_PRESTAMO) references ventas(N_PRESTAMO));
```

Script de creación de tablas



01-CREA_TABLAS.
sql

Creación de tablas (Cont.)

producto Query 1 caida_prestamos costos incobrabilidad pagos

1 • SELECT * FROM ent_fciera.producto;

Result Grid Filter Rows: Edit: Export/Import

PRODUCTO_ID	NOMBRE_PRODUCTO
1	Pago Voluntario
2	Emplados
3	Jubilados
4	Empresas
* NULL	NULL

caida_prestamos pagos

1 • SELECT * FROM ent_fciera.caida_prestamos;

Result Grid Filter Rows: Edit: Export/Import

N_PRESTAMO	N_CUOTA	F_VTO	CAPITAL	INTERES	IVA
1	1	2024-04-01	4228.45	9641.87	2024.79
1	2	2024-05-01	4721.77	9181.19	1928.05
1	3	2024-06-01	5272.64	8676.08	1821.98
1	4	2024-07-01	5887.78	8122.28	1705.68
1	5	2024-08-01	6574.69	7515.08	1578.17
1	6	2024-09-01	7341.74	6849.33	1438.36
1	7	2024-10-01	8198.27	6119.39	1285.07
1	8	2024-11-01	9154.74	5319.07	1117
1	9	2024-12-01	10222.8	4441.58	932.73
1	10	2025-01-01	11415.5	3479.49	730.69
1	11	2025-02-01	12747.2	2424.64	509.17
1	12	2025-03-01	14234.4	1268.08	266.3

Query 1 costos caida_prestamos incobrabilidad pagos

1 • SELECT * FROM ent_fciera.costos;

Result Grid Filter Rows: Edit: Export/Import

PRODUCTO_ID	PERIODO	CONCEPTO	PCIO_PROM_UNIT
1	2024-03	Comercializador	1000
1	2024-04	Comercializador	1200
1	2024-03	Cuponera	1000
4	2024-03	Cuponera	1000
1	2024-04	Cuponera	1020
4	2024-04	Cuponera	1020
1	2024-03	Seguro costo fijo	300
2	2024-03	Seguro costo fijo	200
3	2024-03	Seguro costo fijo	300
4	2024-03	Seguro costo fijo	300
1	2024-04	Seguro costo fijo	300
2	2024-04	Seguro costo fijo	200

caida_prestamos pagos

1 • SELECT * FROM ent_fciera.ventas;

Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Contents

N_CLIENTE	N_PRESTAMO	FECHA_LIQUIDACION	PRODUCTO_ID	PLAZO	TNA	CAP_FINANCIADO	ATRASO	F_LIT_YTO_IMPAGO
500	1	2024-03-01	1	12	1.4	100000	1	2024-05-01
35	2	2024-03-10	3	24	1.5	750000	0	2024-06-10
15	3	2024-03-10	1	18	1.4	300000	0	2024-05-10
80	4	2024-03-25	2	12	1.2	250000	0	2024-06-25
3	5	2024-03-26	4	18	1.3	800000	6	2024-04-26
5	6	2024-04-02	3	36	1.5	1000000	0	2024-05-02
10	7	2024-04-05	1	18	1.4	250000	0	2024-05-05
15	8	2024-04-10	4	12	1.3	980000	0	2024-05-10
5	9	2024-04-11	2	336	1.2	8202000	0	2024-06-11
79	10	2024-04-11	2	12	1.2	253000	0	2024-05-11
* NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Query 1 incobrabilidad caida_prestamos pagos

1 • SELECT * FROM ent_fciera.incobrabilidad;

Result Grid Filter Rows: Edit: Export/Import

ATRASO	PORCENT_INCOB
0	0.01
1	0.01
2	0.01
3	0.01
4	0.01
5	0.01
6	0.01
7	0.01
8	0.01
9	0.01
10	0.01
11	0.01

caida_prestamos pagos

1 • SELECT * FROM ent_fciera.pagos;

Result Grid Filter Rows: Edit: Export/Import

PERIODO	COMP_PAGO	N_PRESTAMO	N_CUOTA	F_PAGO	IMPORTE
2024-04	1	1	1	2024-04-01	15895
2024-04	2	2	1	2024-04-10	99649
2024-04	3	2	2	2024-04-10	99377
2024-04	4	4	2	2024-04-15	36563
2024-04	5	3	1	2024-04-23	40566
2024-04	6	4	1	2024-04-23	36691
2024-04	7	7	1	2024-04-25	33805
2024-04	8	9	1	2024-04-25	516715
* NULL	NULL	NULL	NULL	NULL	NULL

Inserción de datos

Los datos en las tablas se ingresaron siguiendo el siguiente orden:

- ▶ Mediante script para las tablas:
 1. Tabla Producto
 2. Tabla Costos
 3. Tabla Incobrabilidad
- ▶ Mediante archivos '.csv' en el siguiente orden:
 1. Tabla Ventas
 2. Tabla caída_prestamos
 3. Tabla pagos

Script de inserción de datos



02-INSERTA_DAT
OS_TABLAS

Archivos para carga de datos



01-VENTAS_CARG
A.csv



02-CAIDA_CARGA

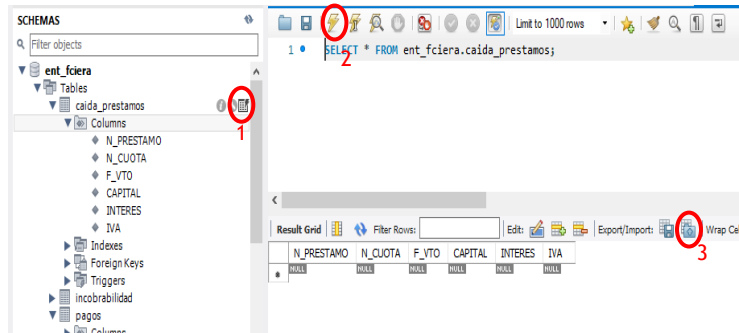


03-PAGOS_CARG
A.csv

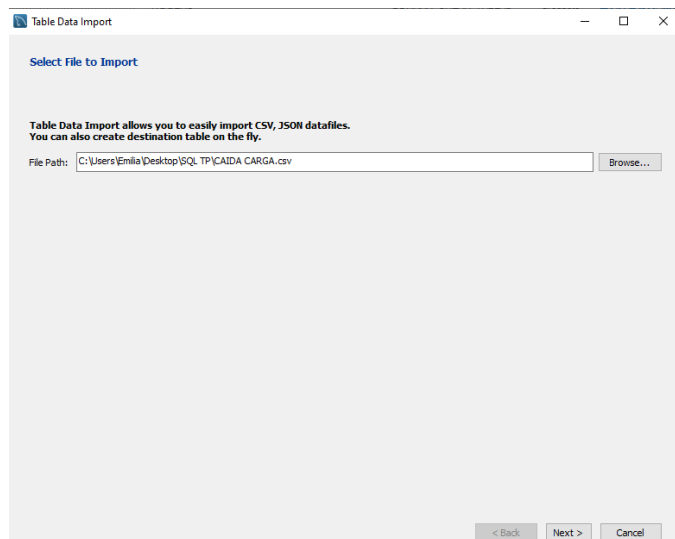
Para los tres casos el procedimiento realizado fue el siguiente:

Inserción de datos (Cont.)

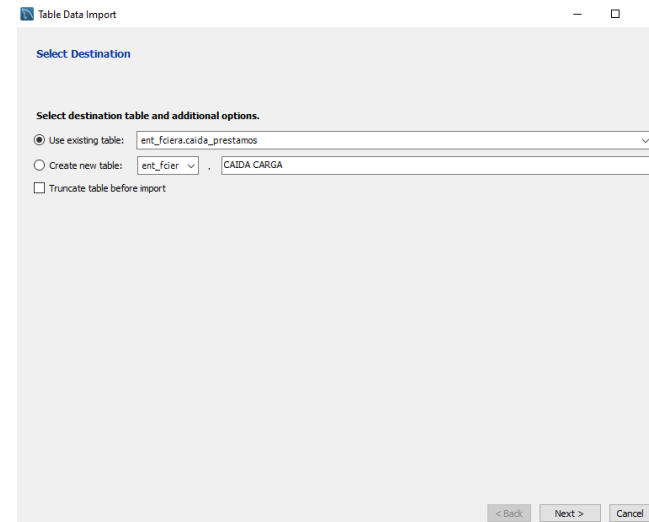
1. Se creo la consulta de la tabla mediante el ultimo icono que se encuentra al costado del nombre de esta y se ejecuta la sentencia. Luego se comenzó el proceso de importación de la información con el Icono marcado con el número 3.



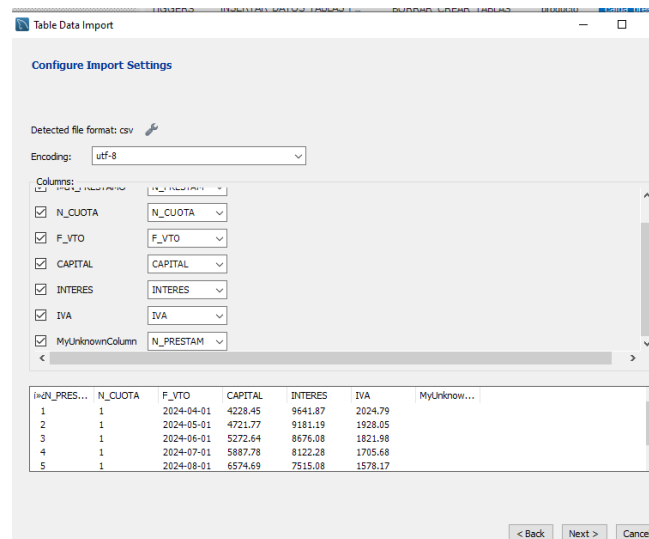
2. Se selecciono el archivo a subir



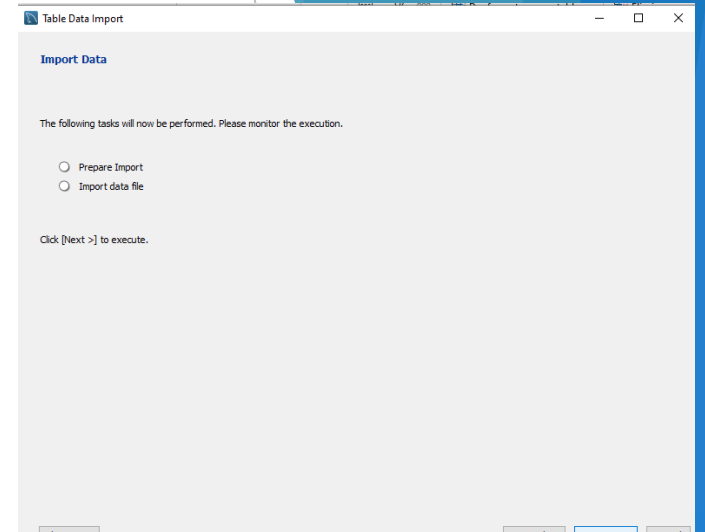
3. Se marco el uso de una tabla existente y se seleccionó la tabla destino de los datos a importar



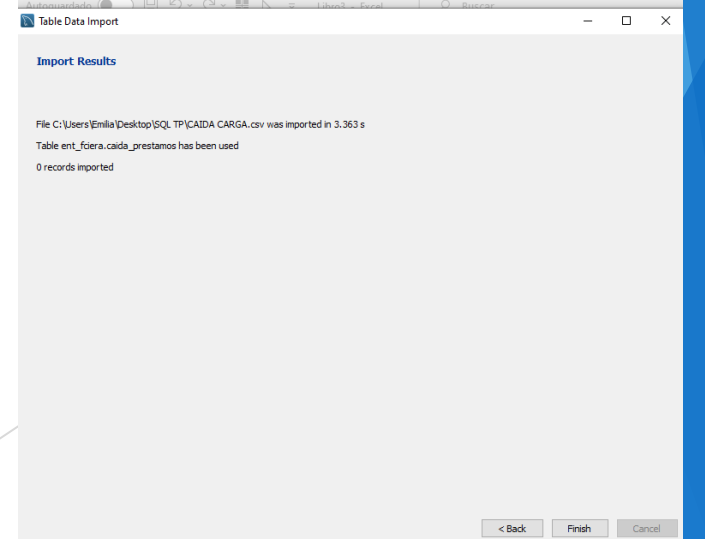
4. Las columnas del archivo poseen idéntico nombre a los de la tabla por lo que la relación entre el campo a subir y el destino fue automática



5. Se procedió a ejecutar la carga



5. Y luego de controlar que se hayan cargado todos los registros se procedió a finalizar la carga



Stored Procedures

► STORE PROCEDURE #1:

OBJETIVO: actualizar la columna de ultimo vencimiento impago en la tabla de ventas este dato se actualiza con cada pago que se realiza. En el caso de un negocio real donde diariamente se realizan pagos, este stored debería ejecutarse de forma diaria

SENTENCIA:

```
USE `ent_fciera`;
DROP procedure IF EXISTS `01-ACT_F_VTA_TABLA_VTA`;
DELIMITER $$
USE `ent_fciera`$$
CREATE PROCEDURE `01-ACT_F_VTA_TABLA_VTA` ()
BEGIN
create temporary table TEMP_PROX_VTO
Select  c.N_PRESTAMO,  min(c.F_VTO) as PROX_VTO  from caida_prestamos c
where concat(c.N_PRESTAMO, '_',c.N_CUOTA) not in (select concat(p.N_PRESTAMO, '_',p.N_CUOTA) from pagos p)
group by (c.N_PRESTAMO);
update ventas v, TEMP_PROX_VTO p
set v.F_ULT_VTO_IMPAGO=p.PROX_VTO
where v.N_PRESTAMO=P.N_PRESTAMO;
drop temporary table if exists TEMP_PROX_VTO;
END$$
DELIMITER ;
```

► STORE PROCEDURE #2:

OBJETIVO: actualizar la columna de atraso en la tabla de ventas, al igual que la anterior, este dato es de actualización diaria.

SENTENCIA:

```
USE `ent_fciera`;
DROP procedure IF EXISTS `02-ACT_ATR_TABLA_VENTAS`;
DELIMITER $$
USE `ent_fciera`$$
CREATE PROCEDURE `02-ACT_ATR_TABLA_VENTAS` ()
BEGIN
drop temporary table if exists TEMP_ATRASO;
create temporary table TEMP_ATRASO
Select  V.N_PRESTAMO,  IF(datediff(curdate(),v.F_ULT_VTO_IMPAGO)<0,0,datediff(curdate(),v.F_ULT_VTO_IMPAGO)) AS ATR
from ventas v;
update ventas v, TEMP_ATRASO a
set v.atraso=a.atr
where v.N_PRESTAMO=a.N_PRESTAMO;
END$$
DELIMITER ;
```

1 • SELECT * FROM ent_fciera.ventas;

N_CLIENTE	N_PRESTAMO	FECHA_LIQUIDACION	PRODUCTO_ID	PLAZO	TNA	CAP_FINANCIADO	ATRASO	F_ULT_VTO_IMPAGO
500	1	2024-03-01	1	12	1.4	100000	1	2024-05-01
35	2	2024-03-10	3	24	1.5	750000	0	2024-06-10
15	3	2024-03-10	1	18	1.4	300000	0	2024-05-10
80	4	2024-03-25	2	12	1.2	250000	0	2024-06-05
3	5	2024-03-26	4	18	1.3	800000	6	2024-04-26
5	6	2024-04-02	3	36	1.5	1000000	0	2024-05-02
10	7	2024-04-05	1	18	1.4	250000	0	2024-06-05
15	8	2024-04-10	4	12	1.3	980000	0	2024-05-10
5	9	2024-04-11	2	336	1.2	8202000	0	2024-06-11
79	10	2024-04-11	2	12	1.2	253000	0	2024-05-11

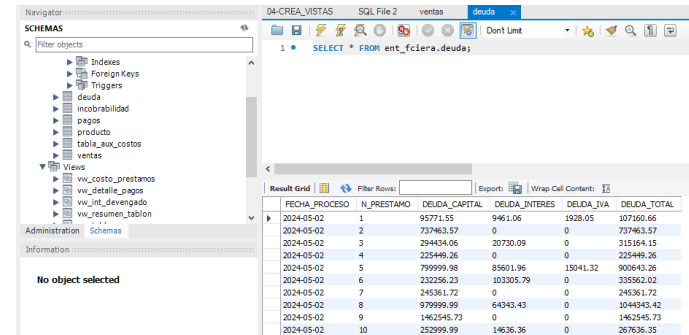
Stored Procedures

► STORE PROCEDURE #3:

OBJETIVO: Creación de una nueva tabla que contenga la deuda de los clientes, este stored se debe ejecutar post creación de las vistas.

SENTENCIA:

```
USE `ent_fciera`;
DROP procedure IF EXISTS `03-TABLA_DEUDA`;
DELIMITER $$
USE `ent_fciera`$$
CREATE PROCEDURE `03-TABLA_DEUDA` ()
BEGIN
drop table if exists DEUDA;
create table DEUDA
select curdate() as FECHA_PROCESO,d.N_PRESTAMO,round(SUM(d.CAPITAL),2) AS DEUDA_CAPITAL,round(SUM(d.INT_DEV),2) AS
DEUDA_INTERES,round(SUM(d.IVA_DEV),2) AS DEUDA_IVA,round(SUM(d.CAPITAL) + SUM(d.INT_DEV) + SUM(d.IVA_DEV),2) AS DEUDA_TOTAL
from VW_INT_DEVENGADO d
where d.ESTADO_CUOTA!='CANCELADA'
group by d.N_PRESTAMO;END$$DELIMITER ;
```



The screenshot shows a database management tool interface. On the left, a 'SCHEMAS' pane lists various database objects like 'Indices', 'Foreign Keys', 'Triggers', 'deuda', 'incobrabilidad', 'pagos', 'productos', 'tabla_aux_costos', 'ventas', and 'Views'. The 'Views' section is expanded, showing 'vw_costo_prestamos', 'vw_detalle_pagos', 'vw_int_devengado', and 'vw_resumen_tablon'. The main window displays a SQL query: 'SELECT * FROM ent_fciera.deuda;'. Below the query, a 'Result Grid' shows the output of the query, which is a table with 6 columns: 'FECHA_PROCESO', 'N_PRESTAMO', 'DEUDA_CAPITAL', 'DEUDA_INTERES', 'DEUDA_IVA', and 'DEUDA_TOTAL'. The table contains 10 rows of data for the date '2024-05-02'.

FECHA_PROCESO	N_PRESTAMO	DEUDA_CAPITAL	DEUDA_INTERES	DEUDA_IVA	DEUDA_TOTAL
2024-05-02	1	95771.55	9461.06	1928.05	107160.66
2024-05-02	2	737463.57	0	0	737463.57
2024-05-02	3	294434.06	20730.09	0	315164.15
2024-05-02	4	225449.26	0	0	225449.26
2024-05-02	5	799999.98	85601.96	15041.32	900643.26
2024-05-02	6	232256.23	103305.79	0	335562.02
2024-05-02	7	245361.72	0	0	245361.72
2024-05-02	8	979999.99	64343.43	0	1044343.42
2024-05-02	9	1462545.73	0	0	1462545.73
2024-05-02	10	252999.99	14636.36	0	267636.35

Script de creación de stored procedures

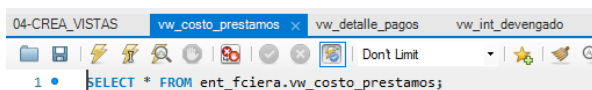


03-CREA_STORE_
PROCEDURES.sql

Views

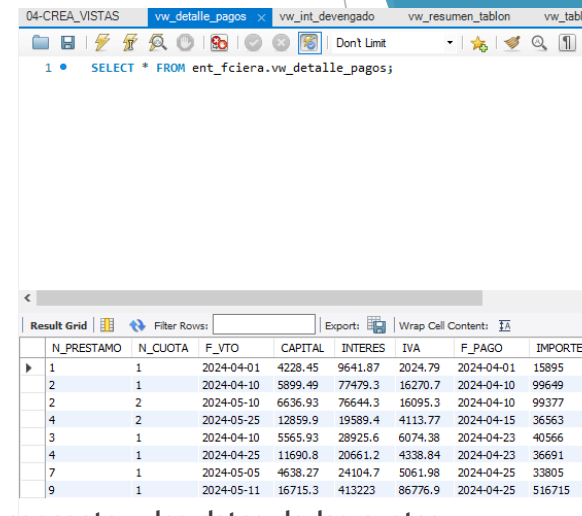
- **VISTA #1 - VW_DETALLE_PAGOS:** muestra el detalle por concepto y los datos de las cuotas pagas. Las tablas que la integran son Caída_prestamos y pagos. Se utiliza para conocer en detalle los pagos realizados. La sentencia para su creación es la siguientes:

```
drop view if exists `VW_DETALLE_PAGOS`;
create view `VW_DETALLE_PAGOS` as
select c.N_PRESTAMO,c.N_CUOTA,c.F_VTO,c.CAPITAL,c.INTERES,c.IVA,p.F_PAGO,
p.IMPORTE from caida_prestamos c inner join pagos p on p.N_PRESTAMO=c.N_PRESTAMO
and p.N_CUOTA=c.N_CUOTA;
```



04-CREA_VISTAS vw_costo_prestamos vw_detalle_pagos vw_int_devengado

1 • SELECT * FROM ent_fciera.vw_detalle_pagos;



04-CREA_VISTAS vw_detalle_pagos vw_int_devengado vw_resumen_tablon vw_tablon

1 • SELECT * FROM ent_fciera.vw_detalle_pagos;

	N_PRESTAMO	N_CUOTA	F_VTO	CAPITAL	INTERES	IVA	F_PAGO	IMPORTE
1	1	1	2024-04-01	4228.45	9641.87	2024.79	2024-04-01	15895
2	1	1	2024-04-10	5899.49	77479.3	16270.7	2024-04-10	99649
2	2	2	2024-05-10	6636.93	76644.3	16095.3	2024-04-10	99377
4	2	2	2024-05-25	12859.9	19589.4	4113.77	2024-04-15	36563
3	1	1	2024-04-10	5565.93	28925.6	6074.38	2024-04-23	40566
4	1	1	2024-04-25	11690.8	20661.2	4338.84	2024-04-23	36691
7	1	1	2024-05-05	4638.27	24104.7	5061.98	2024-04-25	33805
9	1	1	2024-05-11	16715.3	413223	86776.9	2024-04-25	516715

- **VISTA #2- VW_COSTO_PRESTAMOS :** muestra el detalle por concepto y los datos de las cuotas pagas. Las tablas que la integran son ventas, costos y producto. Se utiliza para conocer en detalle los pagos realizados. La sentencia para su creación es la siguientes:

```
drop view if exists `VW_COSTO_PRESTAMOS`;
create view `VW_COSTO_PRESTAMOS` as
select c.PERIODO,v.N_PRESTAMO,p.NOMBRE_PRODUCTO,sum(c.PCIO_PROM_UNIT) as COSTO
from ventas v
left join costos c on v.PRODUCTO_ID=c.PRODUCTO_ID and
c.PERIODO=concat(year(v.FECHA_LIQUIDACION),if(length(month(v.FECHA_LIQUIDACION))=1,
concat('0',month(v.FECHA_LIQUIDACION)),month(v.FECHA_LIQUIDACION)))
left join producto p on p.PRODUCTO_ID=c.PRODUCTO_ID
group by v.N_PRESTAMO,c.PERIODO,p.NOMBRE_PRODUCTO;
```



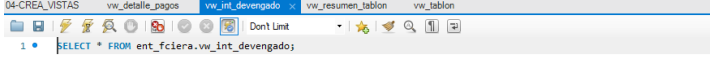
Result Grid Filter Rows: Export: Wrap Cell Content:

	PERIODO	N_PRESTAMO	NOMBRE_PRODUCTO	COSTO
►	202403	1	Pago Voluntario	2300
	202403	2	Jubilados	300
	202403	3	Pago Voluntario	2300
	202403	4	Emplados	200
	202403	5	Empresas	1300
	202404	6	Jubilados	300
	202404	7	Pago Voluntario	2520
	202404	8	Empresas	1320
	202404	9	Emplados	200
	202404	10	Emplados	200

- **VISTA #3 - VW_INT_DEVENGADO:** muestra interés devengado a la fecha de la ejecución, así como también el estado de la cuota . La integran la tabla caída_prestamos y la vista VW_DETALLE_PAGOS. Se utiliza para conocer el estado de cada cuota de los clientes y poder realizar diversos análisis sobre ellos.

Views (Cont.)

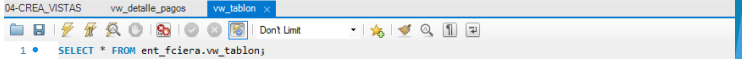
```
drop view if exists `VW_INT_DEVENGADO`;
create view `VW_INT_DEVENGADO` as
select  CURDATE() AS FECHA_PROCESO,c.N_PRESTAMO,c.N_CUOTA,date_add(c.F_VTO,interval(-1) month) as
F_INICIO,c.F_VTO,p.F_PAGO,c.CAPITAL,
case
when curdate()>c.F_VTO then round(c.INTERES,2)
when curdate()>p.F_PAGO then round(c.INTERES,2)
when curdate()< date_add(c.F_VTO,interval(-1) month) then 0
else round(c.INTERES/datediff(c.F_VTO,date_add(c.F_VTO,interval(-1) month))*datediff(curdate(),
date_add(c.F_VTO,interval(-1) month)),2)
end as INT_DEV,
case
when curdate()>c.F_VTO then round(c.IVA,2)
when curdate()>p.F_PAGO then round(c.IVA,2)
else 0
end as IVA_DEV,
case
when p.F_PAGO is not null then 'CANCELADA'
when curdate()>c.F_VTO then 'VENCIDA'
when curdate()< date_add(c.F_VTO,interval(-1) month) then 'NO VIGENTE' else 'VIGENTE' end as ESTADO_CUOTA,case
when p.F_PAGO is not null then 0
when curdate()>c.F_VTO then datediff(curdate(),c.F_VTO)
else 0
end as ATRASO_CUOTA
from caida_prestamos c
left join VW_DETALLE_PAGOS p on p.N_PRESTAMO=c.N_PRESTAMO and p.N_CUOTA=c.N_CUOTA;
```



04-CREA_VISTAS vw_detalle_pagos vw_int_devengado vw_resumen_tablon vw_tablon

1 • SELECT * FROM ent_fciera.vw_int_devengado;

FECHA_PROCESO	N_PRESTAMO	N_CUOTA	F_INICIO	F_VTO	F_PAGO	CAPITAL	INT_DEV	IVA_DEV	ESTADO_CUOTA	ATRASO_CUOTA
2024-05-02	1	1	2024-03-01	2024-04-01	2024-04-01	4228.45	9641.87	2024.79	CANCELADA	0
2024-05-02	1	2	2024-04-01	2024-05-01	2024-05-01	4721.77	9181.19	1928.05	VENCIDA	1
2024-05-02	1	3	2024-05-01	2024-06-01	2024-06-01	5272.64	279.87	0	VIGENTE	0
2024-05-02	1	4	2024-06-01	2024-07-01	2024-07-01	5807.78	0	0	NO VIGENTE	0
2024-05-02	1	5	2024-07-01	2024-08-01	2024-08-01	6574.69	0	0	NO VIGENTE	0
2024-05-02	1	6	2024-08-01	2024-09-01	2024-09-01	7341.74	0	0	NO VIGENTE	0
2024-05-02	1	7	2024-09-01	2024-10-01	2024-10-01	8198.27	0	0	NO VIGENTE	0
2024-05-02	1	8	2024-10-01	2024-11-01	2024-11-01	9154.74	0	0	NO VIGENTE	0



04-CREA_VISTAS vw_detalle_pagos vw_tablon

1 • SELECT * FROM ent_fciera.vw_tablon;

FECHA_PROCESO	N_PRESTAMO	NOMBRE_PRODUCTO	FECHA_LIQUIDACION	PLAZO	TNA	CAP_FINANCIADO	DEUDA_CAPITAL	DEUDA_INTERES	OTRO
2024-05-02	1	Pago Voluntario	2024-03-01	12	1.4	100000	252999.99	14636.36	0
2024-05-02	1	Pago Voluntario	2024-03-01	12	1.4	100000	252999.99	14636.36	0
2024-05-02	1	Pago Voluntario	2024-03-01	12	1.4	100000	252999.99	14636.36	0
2024-05-02	1	Pago Voluntario	2024-03-01	12	1.4	100000	252999.99	14636.36	0
2024-05-02	1	Pago Voluntario	2024-03-01	12	1.4	100000	252999.99	14636.36	0
2024-05-02	1	Pago Voluntario	2024-03-01	12	1.4	100000	1462545.73	0	0
2024-05-02	1	Pago Voluntario	2024-03-01	12	1.4	100000	1462545.73	0	0
2024-05-02	1	Pago Voluntario	2024-03-01	12	1.4	100000	1462545.73	0	0
2024-05-02	1	Pago Voluntario	2024-03-01	12	1.4	100000	1462545.73	0	0
2024-05-02	1	Pago Voluntario	2024-03-01	12	1.4	100000	1462545.73	0	0
2024-05-02	1	Pago Voluntario	2024-03-01	12	1.4	100000	1462545.73	0	0
2024-05-02	1	Pago Voluntario	2024-03-01	12	1.4	100000	979999.99	64343.43	0

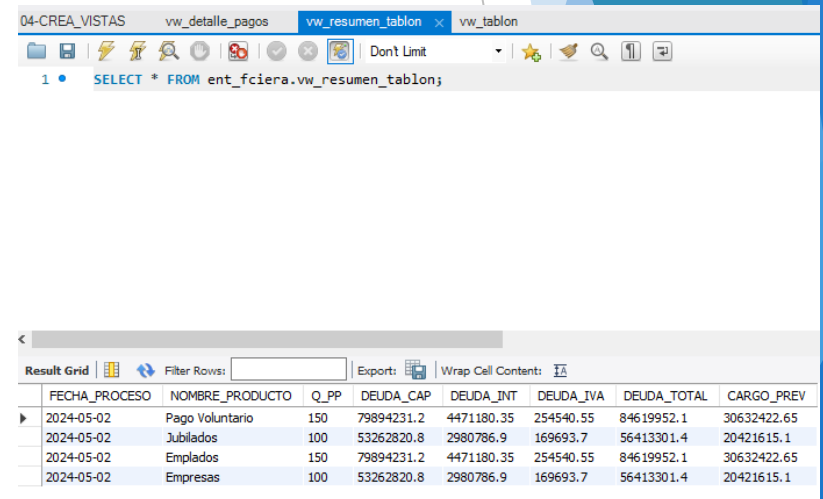
- **VISTA #4 - VW_TABLON:** recopila toda la información del préstamo en un solo informe. Para el armado de esta se involucran las tablas: ventas, producto, deuda, incobrabilidad y la vista vw_costo_prestamos. La sentencia para su creación es la siguientes:

Views (Cont.)

```
drop view if exists `VW_TABLON`;
create view `VW_TABLON` as
select distinct
CURDATE() AS FECHA_PROCESO,h v.N_PRESTAMO, p.NOMBRE_PRODUCTO, v.FECHA_LIQUIDACION, v.PLAZO,v.TNA,
v.CAP_FINANCIADO, d.DEUDA_CAPITAL, d.DEUDA_INTERES, d.DEUDA_IVA, d.DEUDA_TOTAL, v.ATRASO,
round(d.DEUDA_TOTAL * I.PORCENT_INCOB,2) as CARGO_INCOB, c.PERIODO as PERIODO_COSTO, c.COSTO
from ventas v
left join VW_COSTO_PRESTAMOS c on c.N_PRESTAMO=v.N_PRESTAMO
left join DEUDA d on c.N_PRESTAMO=v.N_PRESTAMO
left join incobrabilidad i on v.ATRASO=V.ATRASO
left join producto p on p.PRODUCTO_ID=v.PRODUCTO_ID;
```

- **VISTA #5 - VW_RESUMEN TABLON** : presenta un resumen del tablón agrupado por producto. Para vista solo se utilizó la vista VW_TABLON. La sentencia para su creación es la siguientes:

```
drop view if exists `VW_RESUMEN_TABLON`;
create view `VW_RESUMEN_TABLON` as
select distinct
t.FECHA_PROCESO, t.NOMBRE_PRODUCTO, round(count(t.N_PRESTAMO),0) as Q_PP,
round(sum(t.DEUDA_CAPITAL),2) as DEUDA_CAP, round(sum(t.DEUDA_INTERES),2) as DEUDA_INT,
round(sum(t.DEUDA_IVA),2) as DEUDA_IVA, round(sum(t.DEUDA_TOTAL),2) as DEUDA_TOTAL,
round(sum(t.CARGO_INCOB),2) as CARGO_PREV
from vw_tablon t
where t.DEUDA_TOTAL!=0
group by (t.NOMBRE_PRODUCTO);
```



The screenshot shows a database management interface with two tabs: 'vw_detalle_pagos' and 'vw_resumen_tablon'. The 'vw_resumen_tablon' tab is active, displaying a SQL query: `1 • SELECT * FROM ent_fciera.vw_resumen_tablon;` Below the query, a 'Result Grid' shows the data for the view. The data is grouped by product name, showing the date of the process, the product name, the quantity of products (Q_PP), and the various debt components (DEUDA_CAP, DEUDA_INT, DEUDA_IVA, DEUDA_TOTAL) and the previous cargo (CARGO_PREV).

FECHA_PROCESO	NOMBRE_PRODUCTO	Q_PP	DEUDA_CAP	DEUDA_INT	DEUDA_IVA	DEUDA_TOTAL	CARGO_PREV
2024-05-02	Pago Voluntario	150	79894231.2	4471180.35	254540.55	84619952.1	30632422.65
2024-05-02	Jubilados	100	53262820.8	2980786.9	169693.7	56413301.4	20421615.1
2024-05-02	Emplados	150	79894231.2	4471180.35	254540.55	84619952.1	30632422.65
2024-05-02	Empresas	100	53262820.8	2980786.9	169693.7	56413301.4	20421615.1

Script de creación de vistas



04-CREA_VISTAS.s

ql

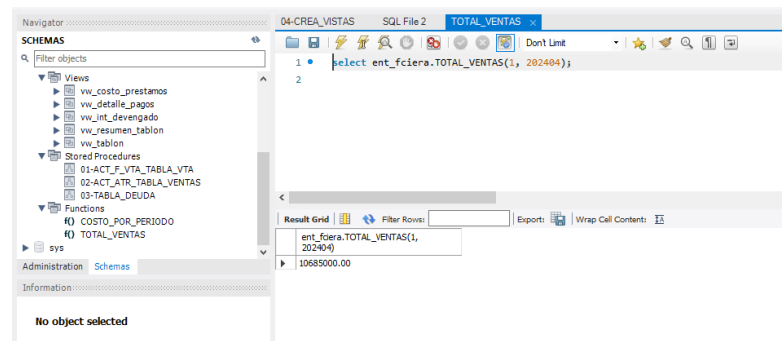
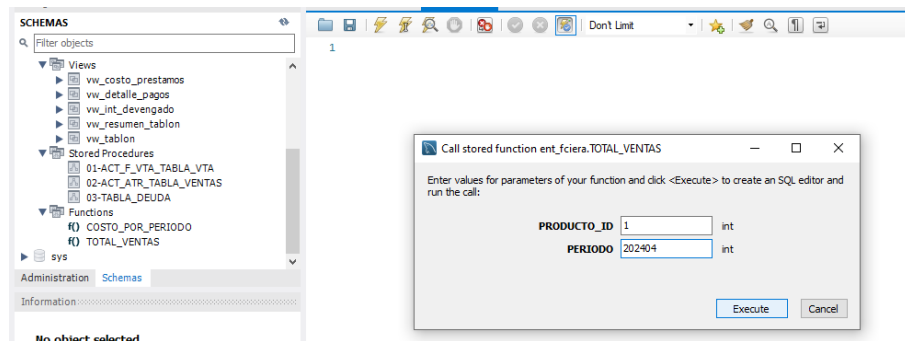
Funciones

► FUNCIÓN #1:

OBJETIVO: esta función nos permite conocer el total de las ventas por periodo y producto, podrá ser utilizada por cada dueño de producto podrá conocer rápidamente la producción del mes deseado

SENTENCIA:

```
USE `ent_fciera`;
DROP function IF EXISTS `TOTAL_VENTAS`;
DELIMITER $$
USE `ent_fciera`$$
create function `TOTAL_VENTAS` (PRODUCTO_ID int, PERIODO int)
returnS decimal (20,2) deterministic
Begin
declare TOTAL decimal(20,2);
select sum(CAP_FINANCIADO) into TOTAL
from ventas where PRODUCTO_ID=PRODUCTO_ID
And
PERIODO=concat(year(FECHA_LIQUIDACION),if(length(month(FECHA_LIQUIDACION))=1,concat('0',month(FECHA_LIQUIDACION)),month(FECHA_LIQUIDACION)));
return TOTAL;
end;$$
DELIMITER ;
```



Funciones (Cont)

► FUNCIÓN #2:

OBJETIVO: determinar el costo total por periodo de alta, el costo se da por única vez al momento del alta por lo que esta función nos permite conocer el total del gasto por mes

SENTENCIA:

```
USE `ent_fciera`;  
DROP function IF EXISTS `COSTO_POR_PERIODO`;  
DELIMITER $$  
USE `ent_fciera` $$  
create function `COSTO_POR_PERIODO` (PERIODO int)  
returnS decimal (20,2) deterministic  
Begin  
declare TOTAL decimal(20,2);  
select sum(COSTO) into TOTAL  
from vw_costo_prestamos where PERIODO=PERIODO;  
return TOTAL;  
end;$$  
DELIMITER ;
```

Script de creación de funciones



05-CREA_FUNCIO
NES.sql

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows the 'ent_fciera' database structure, including Views, Stored Procedures, and Functions. The 'Functions' folder is expanded, showing the 'COSTO_POR_PERIODO' function. In the center, a dialog box titled 'Call stored function ent_fciera.COSTO_POR_PERIODO' prompts the user to enter values for parameters. The 'PERIODO' parameter is set to 202403. On the right, the 'SQL File 2' window shows the SQL script for creating the function. Below the script, the 'Result Grid' displays the output of the function call: 'ent_fciera.COSTO_POR_PERIODO(202403)' returning 10940.00.

Triggers

OBJETIVO: registrar todas las modificaciones que se realizan sobre la tabla costos. Para poder visualizar estos cambios se procedió a crear la tabla 'aux_costos' que contiene todas las modificaciones realizadas sobre el concepto y el costo unitario. Se podrá ver en detalle tanto el dato viejo como el nuevo ya sea por ingreso de nuevos registros, modificaciones o eliminación.

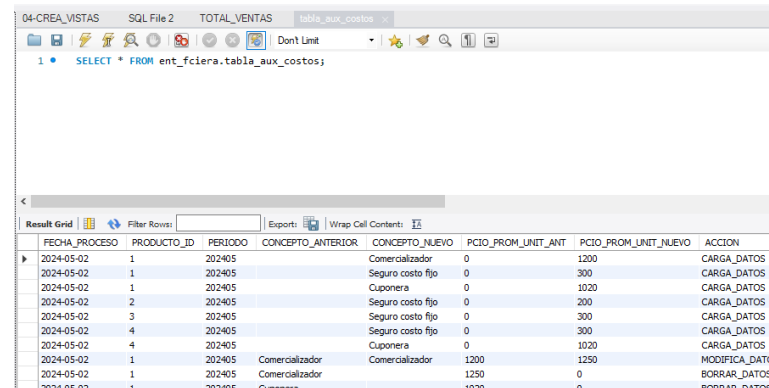
Sentencia para creación de tabla 'aux_costos'

```
drop table if exists tabla_aux_costos;
```

```
create table tabla_aux_costos
(FECHA_PROCESO date,
PRODUCTO_ID int ,
PERIODO int,
CONCEPTO_ANTERIOR varchar(20),
CONCEPTO_NUEVO varchar(20),
PCIO_PROM_UNIT_ANT float,
PCIO_PROM_UNIT_NUEVO float,
ACCION varchar(20));
```

► TRIGGER #1: LOG_NUEVOS_COSTOS (para nuevos registros)

```
drop trigger if exists LOG_COSTOS_NUEVOS;
delimiter $$
create trigger LOG_COSTOS_NUEVOS
after insert on costos
for each row
insert into tabla_aux_costos
(fecha_proceso,producto_id,periodo,concepto_anterior,concepto_nuevo,pcio_prom_unit_ant,pcio_prom_unit_nuevo,accion)
values (curdate(),new.producto_id,new.periodo,"",new.concepto,0,new.pcio_prom_unit,'CARGA_DATOS');$$
delimiter ;
```



The screenshot shows a SQL IDE window with a query editor and a results grid. The query editor contains the following SQL statement:

```
1 • SELECT * FROM ent_fciera.tabla_aux_costos;
```

The results grid displays the following data:

FECHA_PROCESO	PRODUCTO_ID	PERIODO	CONCEPTO_ANTERIOR	CONCEPTO_NUEVO	PCIO_PROM_UNIT_ANT	PCIO_PROM_UNIT_NUEVO	ACCION
2024-05-02	1	202405	Comercializador	Comercializador	0	1200	CARGA_DATOS
2024-05-02	1	202405	Seguro costo fijo	Seguro costo fijo	0	300	CARGA_DATOS
2024-05-02	1	202405	Cuponera	Cuponera	0	1020	CARGA_DATOS
2024-05-02	2	202405	Seguro costo fijo	Seguro costo fijo	0	200	CARGA_DATOS
2024-05-02	3	202405	Seguro costo fijo	Seguro costo fijo	0	300	CARGA_DATOS
2024-05-02	4	202405	Seguro costo fijo	Seguro costo fijo	0	300	CARGA_DATOS
2024-05-02	4	202405	Cuponera	Cuponera	0	1020	CARGA_DATOS
2024-05-02	1	202405	Comercializador	Comercializador	1200	1250	MODIFICA_DATOS
2024-05-02	1	202405	Comercializador	Comercializador	1250	0	BORRAR_DATOS
2024-05-02	1	202405	Cuponera	Cuponera	1020	0	BORRAR_DATOS

Triggers (Cont.)

▶ TRIGGER #2: LOG_COSTOS_MODIF (para modificación de registros)

```
drop trigger if exists LOG_COSTOS_MODIF;
delimiter $$
create trigger LOG_COSTOS_MODIF
before update on costos
for each row
insert into tabla_aux_costos
.fecha_proceso,producto_id,periodo,concepto_anterior,concepto_nuevo,pcio_prom_unit_ant,pcio_prom_unit_nuevo,accion)
values (curdate(),old.producto_id,old.periodo,old.concepto,new.concepto,old.pcio_prom_unit,new.pcio_prom_unit,'MODIFICA_DATOS');
delimiter ;
```

▶ TRIGGER #3: LOG_COSTOS_BORRA (para eliminación de registros)

```
drop trigger if exists LOG_COSTOS_BORRA;
delimiter $$
create trigger LOG_COSTOS_BORRA
before delete on costos
for each row
insert into tabla_aux_costos
.fecha_proceso,producto_id,periodo,concepto_anterior,concepto_nuevo,pcio_prom_unit_ant,pcio_prom_unit_nuevo,accion)
values (curdate(),old.producto_id,old.periodo,old.concepto,"old.pcio_prom_unit,0,'BORRAR_DATOS');
delimiter ;
```

Script de creación triggers

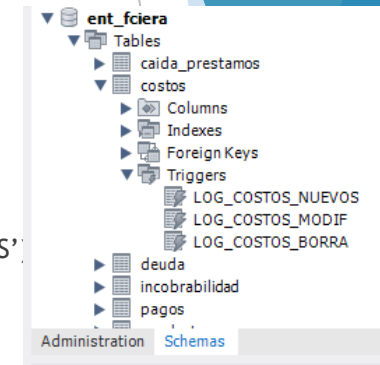


06-CREA_TRIGGER
S.sql

Script de datos para prueba de triggers



07-DATOS_PARA_
RUEBA_TRIGGERS.s



Herramientas y tecnologías usadas

Para el armado del proyecto se utilizaron las siguientes herramientas y tecnologías:

- ▶ MySQL Workbench
- ▶ GitHub
- ▶ Excel
- ▶ PowerPoint
- ▶ Google

Lineas Futuras

Como se comentó en la introducción, el modelo de negocio de una entidad financiera es bastante más complejo al desarrollado. A través del trabajo desarrollado se intentó plasmar el espíritu del mismo con un producto - prestamos.

Enlace a repositorio github

https://github.com/EmiliaPeniaOngania/Segunda_preentrega_SQL.git