**OBSERVACIONES IMPORTANTE ANTES DE INICIAR:**

Cada trabajo o actividad académica que sea desarrollado por usted como evidencia de su proceso de aprendizaje, debe estar 100% libre de frases copiadas desde cualquier fuente original (documento, libro, página web, proyecto de código fuente, video tutorial, etc.) de información.

En ningún momento se le está pidiendo que haga las cosas de 0 y olvide por completo los conocimientos sobre los cuales se basa algún tema de esta asignatura, por el contrario, lo estamos animando a que lea, investigue, comprenda, deduzca, analice, sintetice, concluya, interprete, argumente, domine, practique, critique, niegue o afirme con bases sólidas basadas en la lectura y el estudio autónomo y autodidacta.

Tenga muy presente que mientras estudias esta profesión y durante toda tu vida, siempre seras evaluado y medido de acuerdo por un docente, por un instructor, por tu equipo de trabajo, por tus jefes, por la sociedad o por tus clientes.

Antes de realizar cualquier acción indebida en el desarrollo de esta y cualquier otra actividad académica, recuerda que actualmente existen muchas herramientas modernas y avanzadas que existen analizar, buscar, identificar, rastrear y comparar la idoneidad y originalidad de cada contenidos digital (texto, imágenes, archivos, etc), con especial profundidad en archivos de documentos académico o que contienen fuentes de programas informáticos, las cuales permiten detectar plagio de una manera tremendamente simple, rápida y automatizada.

Evita que el pensamiento del mínimo esfuerzo y el facilismo invada tu mente y tus acciones, ya que este destruirá tus sueños de crecimiento y éxito profesional.

**OBSERVACIONES**:

* **actividad para desarrollar de forma individual**

● Se debe subir al aula virtual de la asignatura por sima-pesad (no se aceptan entregas por otro medio)

● Solo se evaluará y calificará a los estudiantes que hayan subido la actividad por sima-pesad en la fecha estipulada

● Existen muchísimas formas (lógicas y estructurales) de desarrollar un mismo ejercicio de programación, **se evalúa con 1.0 cualquier evidencia de similitud del contenido del código o trabajo con otros contenidos entregados por los alumnos,** **encontrados por internet.** Recuerde que no es ilegal aprender de otros recursos, **lo ilegal es tomar ese recurso para aprovecharse del trabajo hecho por otra persona para luego maquillarlo para tratar de hacerlo pasar por nuestro y ganar una buena calificación,** es decir, **eviden realizar los siguientes trucos viejos y muy fáciles de evidenciar**:

Si la IA te resuelve el los ejercicios, o un amigo o compañero te pasa su tarea, o le pagas a alguien para que la haga por ti o la encuentras resuelta por internet, solo recuerda que en estos momentos ninguno de ellos está perdiendo el tiempo mas que tu, ni actúa de forma más absurda, ilógica e irracional que tu, al realizar tal acción.

**ESTA SON ALGUNAS DE EVIDENCIAS DE PLAGIO O FRAUDE MÁS COMUNES Y FÁCILES DE IDENTIFICAR SIN NECESIDAD DE USAR HERRAMIENTAS TECNOLÓGICAS:**

○ Reorganizar el código, colocando clases, variables y métodos en otros lugares diferentes al lugar donde los dejó el autor original.

○ Cambiar nombres de clases

○ Cambiar nombre de archivos

○ Cambiar nombre del proyecto

○ Cambiar nombres de variables

○ Cambiar el nombre de los métodos o funciones,

○ Cambiar mensajes de salida,

○ Cambiar System.out.print por JOptionPane

○ Cambiar color y apariencia de los componentes de GUI

○ etc

● **Recuerda que:**

* Estudiar y practicar en grupo (cipa) no es sinónimo de entregar el mismo el mismo resultado, que una persona desarrolle el ejercicio y se lo pase a los compañeros de grupo para que lo presenten cuando la tarea es individual.
* Cada estudiante debe tener la capacidad, habilidad, competencia y destreza para poder desarrollar todos los ejercicios de forma individual y con su propia lógica.

**La mejor forma de evitar realizar fraude académico es:**1. Estudiar frecuente y constantemente

2. Identificar prioridades en tu vida y en tu dia a dia

3. Construir un plan de trabajo colocando mayor peso (importancia) a los compromisos que tienen con mayor prioridad para tu vida

4. Ejecuta ese plan dia a dia, evitando posponer, retrasar o aplazar las actividades y compromisos diarios

5. Utiliza internet y sus poderosas herramientas para buscar recursos académicos de los cuales puedas aprender y practicar (Buscadores de recursos académicos, Foros, Libros digitales, cursos en video, tutoriales sobre temas puntuales, etc) y estudiar con base a sus contenidos.

6. Reúnete con tus compañeros (pueden ser de la U, o amigos de barrio, conocidos, etc, que tengan tu misma visión, alto grado de responsabilidad, tus mismas metas y objetivos), esto con el fin de intercambiar conocimientos mutuamente, tratando siempre de aportar y más que preguntar.

7. Pon en práctica los conocimientos adquiridos durante el proceso de estudio, para eso tienes las actividades, pero también puedes diseñar tus propias actividades personales de aprendizaje.

8. Autoevaluate con mucha frecuencia, de esta manera podrás saber como vas, que estas haciendo mal, que estás haciendo bien, si esta lejos o cerca del objetivo.

**FORMATO DE ENTREGA:**

Un archivo comprimido en formato ZIP **(NO USAR OTRO FORMATO) - NO SE ACEPTA**

**Nombre del archivo**:

codigo\_nombre\_apellido\_asignatura\_act\_2\_2023-1**.zip**

**Por ejemplo**:

1234567\_john\_arrieta\_bd1\_act\_2\_2023\_1.**zip**

El archivo comprimido debe contener como mínimo los siguientes archivos.

■ Documento en formato EDITABLE .DOC (NO ES PDF) con el d**esarrollo y explicación detallada** del desarrollo de la actividad.

■ El desarrollo y **entrega del trabajo debe contener todos los ítem** típicos y característicos de un trabajo académico.

1. Presentación

2. Tabla de contenido

3. Introducción

4. Objetivos

5. Justificación

6. Desarrollo

a. Debe ser lo suficientemente explicativo,

b. Argumentativo

c. Ilustrativo

d. Enlace de GitHub del repositorio para descargar los proyectos o archivos de código fuente.

7. Bibliografía

Archivo comprimido con los códigos fuentes del desarrollo de la actividad.

**Normas APA para la elaboración de esta actividad**

<https://www.colconectada.com/normas-apa/>

REQUISITOS:

Para el desarrollo de esta actividad es necesario adjuntar al documento los siguientes recursos:

1. Enunciado del problema
2. Diagrama Entidad Relación
3. Modelo Relacional
4. Script SQL con las sentencias DDL en OracleSQL. PostgreSQL y SQLServer correspondientes a la implementación de la Base de datos
5. Script SQL con las instrucciones DML de tipo Insert con los datos de prueba a insertar en la BD, los cuales son necesario validar la efectividad del desarrollo y resultado de las presuntas prácticas que conforman esta actividad.
6. El desarrollo de cada punto debe incluir:
   1. El texto con la sentencia SQL que da respuesta a la solución de dicha pregunta.
   2. Imágenes con recortes de pantallas en las zonas de los Cliente de BD utilizados para ejecutar y probar cada una de las Instrucción SQL.

**INFORMACIÓN SOBRE LA ACTIVIDAD ACADÉMICA - UNIDAD 2**

De acuerdo a lo estipulado en tutorías sobre lo concerniente a las actividades académicas evaluativas, a continuación se describe el desarrollo de la primera actividad, donde ustedes como equipo deben tomar como base el mismo ejercicio desarrollado en BD-1 (el caso práctico elegido para desarrollar las actividades propuestas en la asignatura BD-1), continuar con el desarrollo de la temática de la asignatura BD-2, por lo que para esta primera actividad (Unidad 1) deben resolver los siguiente puntos sobre Validación de Restricción de inconsistencia de entrada de datos en la BD (Comando Check), Vistas, Transacciones y su aplicación en los motores de bases de datos relacionales OracleSQL. PostgreSQL, SQLServer y MySQL.

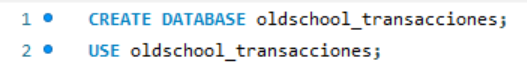
1. Qué es y qué problema resuelve el uso del concepto de Vistas SQL en bases de datos Relacionales o SQL
2. Analizar el Modelo Relacional del ejercicio que seleccionaron como Grupo de BD-1, diseñar una consulta multitabla y probarla (en MySQL, OracleSql, SQLServer y PostgreSQL).
3. Usar la consulta anterior y crear con ella una Vista SQL
4. Mostrar la Vista SQL creada anteriormente
5. Ejecutar la Vista SQL creada anteriormente
6. Abrir la Vista SQL y modificarla para cambiar la consulta por otra consulta o modificar la existente.
7. Volver a ejecutar la Vista SQL
8. Las vistas tienen ventajas, desventajas y limitaciones, muchas de estas características dependen mucho del motor de bases de datos que estemos utilizando, en este punto usted debe:
   1. Describir las ventajas del uso de Vistas SQL para el X motor de bases de datos sobre el cual desarrolló la actividad
   2. Describir las desventajas del uso de Vistas SQL para el X motor de bases de datos sobre el cual desarrolló la actividad
   3. Describir cuales son las limitaciones que presentan las Vistas SQL en el motor de BD que está utilizando, por ejemplo: Cuáles son las instrucciones SQL que no pueden ser utilizadas dentro de una Vista SQL. etc.

1. Para el desarrollo de los siguientes puntos referentes al tema de **Transacciones**, ustedes como grupo deben realizar las siguientes tareas sobre OracleSql, SQLServer y PostgreSQL y MySQL:
   1. Crear una base de datos llamada prueba\_transacciones\_grupoXYZ, siendo XYZ el nombre del ejercicio escogido como equipo.

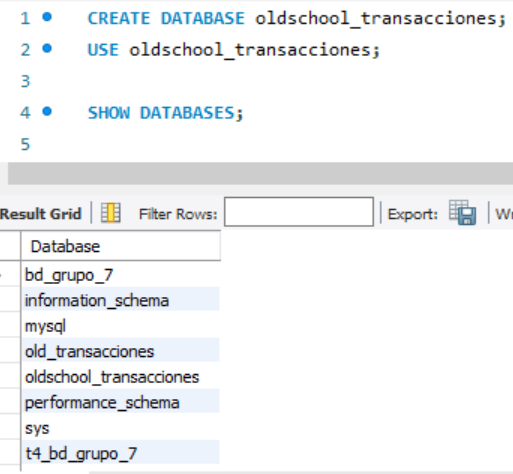
Sentencia SQL

CREATE DATABASE oldschool\_transacciones;

USE oldschool\_transacciones;



SHOW DATABASES;



* 1. Crear tablas **Deudores**(**cc**, clave, nombre, apellido, email). email es único, todos son obligatorios

Sentencia SQL

CREATE TABLE deudores (

cc INT PRIMARY KEY NOT NULL,

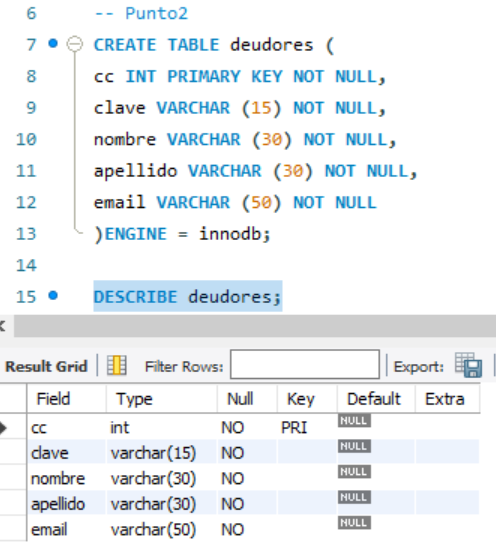
clave VARCHAR (15) NOT NULL,

nombre VARCHAR (30) NOT NULL,

apellido VARCHAR (30) NOT NULL,

email VARCHAR (50) NOT NULL

)ENGINE = innodb;



* 1. Crear tablas **Creditos**(id, fecha, valor, cuotas, interes, estado, deudor\_id), todos son obligatorios, estado por defecto es Activo

Sentencia SQL

CREATE TABLE creditos(

id INT AUTO\_INCREMENT PRIMARY KEY NOT NULL,

fecha DATE NOT NULL,

valor FLOAT NOT NULL,

cuotas INT NOT NULL,

interes\_mes FLOAT NOT NULL,

estado VARCHAR (20) NOT NULL DEFAULT 'Activo',

deudor\_id INT NOT NULL,

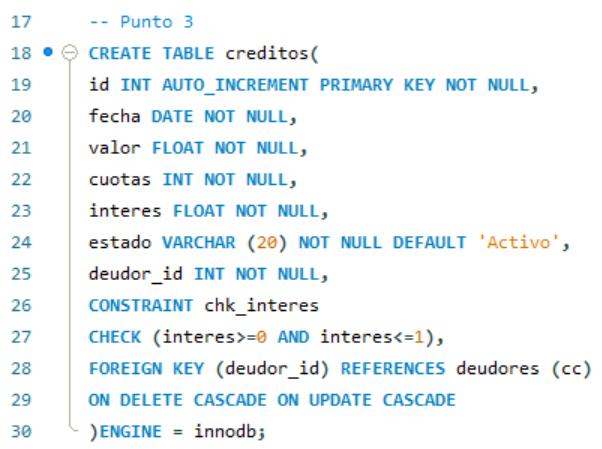
CONSTRAINT chk\_interesmes

CHECK (interes\_mes>=0 AND interes\_mes<=1),

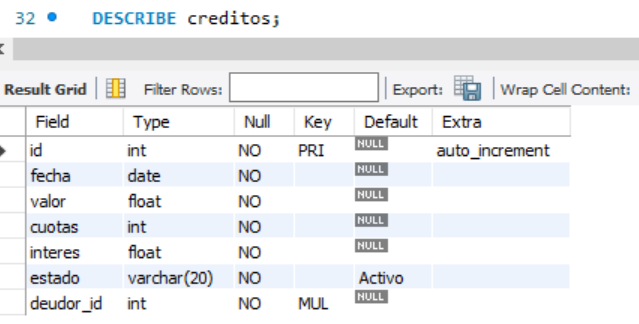
FOREIGN KEY (deudor\_id) REFERENCES deudores (cc)

ON DELETE CASCADE ON UPDATE CASCADE

)ENGINE = innodb;



DESCRIBE creditos;



* 1. Crear tablas **Pagos**(id, fecha, valor, credito\_id). Todos son obligatorios. Use el comando check para evitar que la columna valor solo puede aceptar valor positivos mayores que 0 y la fecha del pago no puede ser mayor que la fecha actual.

Sentencia SQL

CREATE TABLE pagos(

id INT PRIMARY KEY AUTO\_INCREMENT NOT NULL,

fecha DATE NOT NULL,

valor FLOAT NOT NULL ,

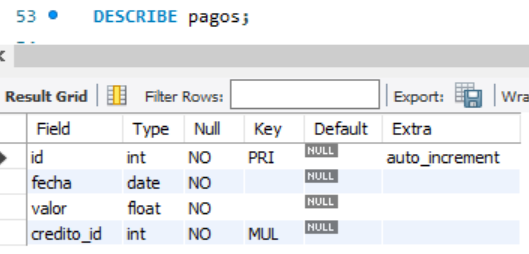
credito\_id INT NOT NULL,

CONSTRAINT chk\_valor\_positivo CHECK(valor>0),

FOREIGN KEY (credito\_id) REFERENCES creditos (id)

ON DELETE CASCADE ON UPDATE CASCADE

)ENGINE = innodb;



DELIMITER $$

CREATE TRIGGER verificar\_fecha BEFORE INSERT ON pagos

FOR EACH ROW

BEGIN

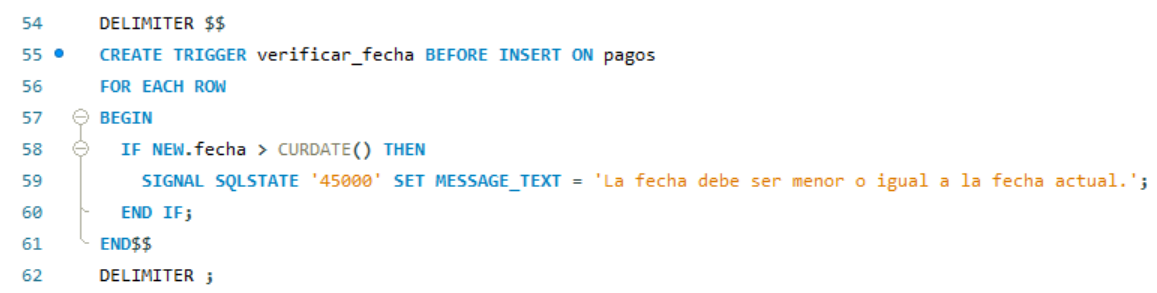
IF NEW.fecha > CURDATE() THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'La fecha debe ser menor o igual a la fecha actual.';

END IF;

END$$

DELIMITER ;



En este caso fue necesario colocar un trigger para restringir a que las fechas ingresadas no sean menores a la fecha actual.

* 1. Insertar al menos un Deudor:

El primero con cc=123, calve=Abc, nombre = Fulanito, apellido=De tal, email=[fulanito1@gmail.com](mailto:fulanito1@gmail.com)}

Sentencia SQL

INSERT INTO deudores

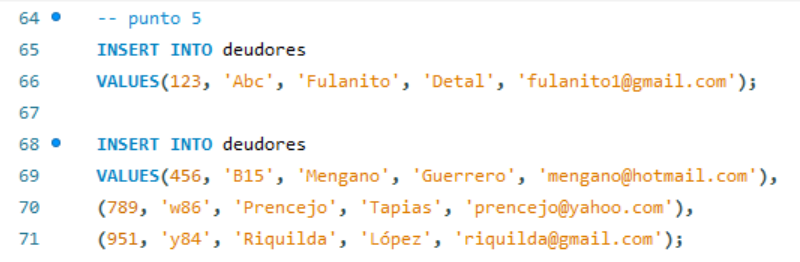
VALUES(123, 'Abc', 'Fulanito', 'Detal', 'fulanito1@gmail.com');

INSERT INTO deudores

VALUES(456, 'B15', 'Mengano', 'Guerrero', 'mengano@hotmail.com'),

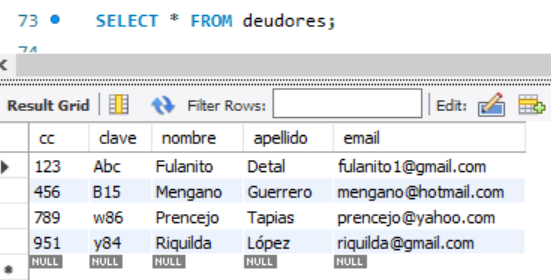
(789, 'w86', 'Prencejo', 'Tapias', 'prencejo@yahoo.com'),

(951, 'y84', 'Riquilda', 'López', 'riquilda@gmail.com');



* 1. Mostrar todos los Deudores insertados en la BD

SELECT \* FROM deudores;



* 1. Insertar al menos un Crédito:

id=1, fecha=hace un año, valor=100000, cuotas=5, interes\_mes=40%, deudor\_id=123

Sentencia SQL

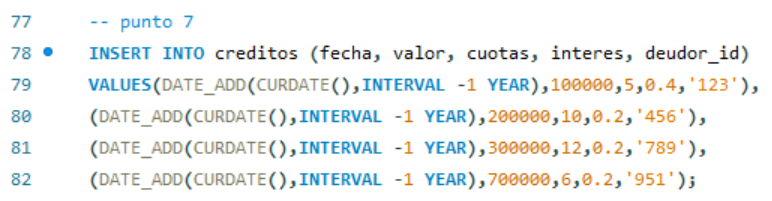
INSERT INTO creditos (fecha, valor, cuotas, interes, deudor\_id)

VALUES(DATE\_ADD(CURDATE(),INTERVAL -1 YEAR),100000,5,0.4,'123'),

(DATE\_ADD(CURDATE(),INTERVAL -1 YEAR),200000,10,0.2,'456'),

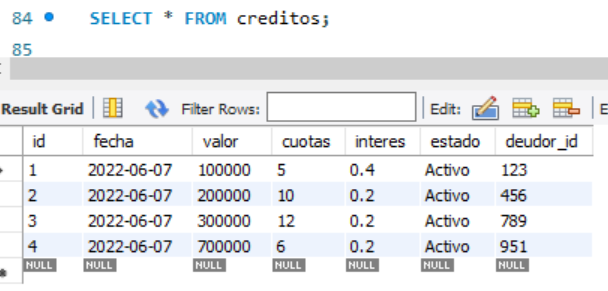
(DATE\_ADD(CURDATE(),INTERVAL -1 YEAR),300000,12,0.2,'789'),

(DATE\_ADD(CURDATE(),INTERVAL -1 YEAR),700000,6,0.2,'951');



* 1. Mostrar todos los Créditos insertados en la BD

SELECT \* FROM creditos;

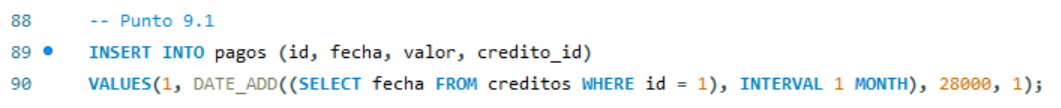


* 1. Insertar 5 pagos así (no modifique ningún valor del ejemplo, colocar las fechas correspondientes, el formato de fecha correspondiente para bd):
     1. id=1, fecha=**un mes después del crédito**, valor=28000, credito\_id=1

Sentencia SQL

INSERT INTO pagos (id, fecha, valor, credito\_id)

VALUES(1, DATE\_ADD((SELECT fecha FROM creditos WHERE id = 1), INTERVAL 1 MONTH), 28000, 1);

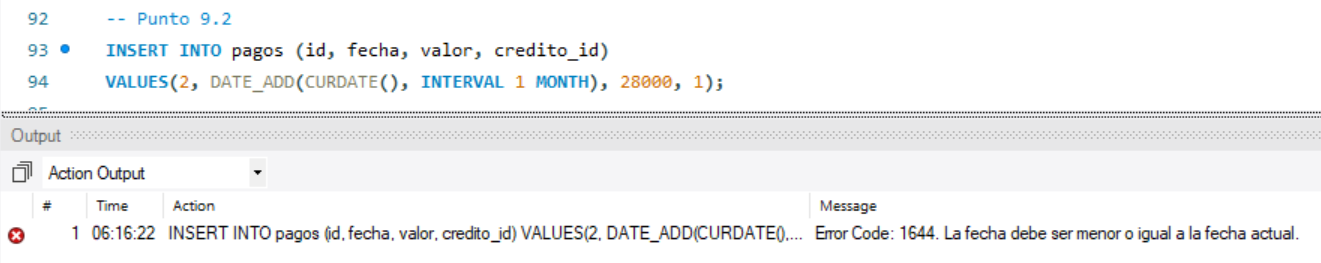


* + 1. id=2, fecha=**un mes después de la fecha actual,** valor=28000, credito\_id=1

Sentencia SQL

INSERT INTO pagos (id, fecha, valor, credito\_id)

VALUES(2, DATE\_ADD(CURDATE(), INTERVAL 1 MONTH), 28000, 1);



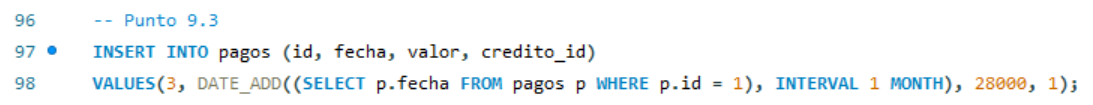
Se observa que al intentar introducir un pago con fecha mayor a la fecha actual da un error esto nos indica que las restricción colocada en un ejercicio anterior es correcta.

* + 1. id=3, fecha=**un mes después del primer pago,** valor=28000, credito\_id=1

Sentencia SQL

INSERT INTO pagos (id, fecha, valor, credito\_id)

VALUES(3, DATE\_ADD((SELECT p.fecha FROM pagos p WHERE p.id = 1), INTERVAL 1 MONTH), 28000, 1);

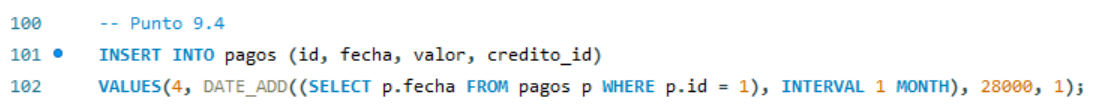


* + 1. id=4, fecha=**un mes después del primer pago,** valor=28000, credito\_id=1

Sentencia SQL

INSERT INTO pagos (id, fecha, valor, credito\_id)

VALUES(4, DATE\_ADD((SELECT p.fecha FROM pagos p WHERE p.id = 1), INTERVAL 1 MONTH), 28000, 1);

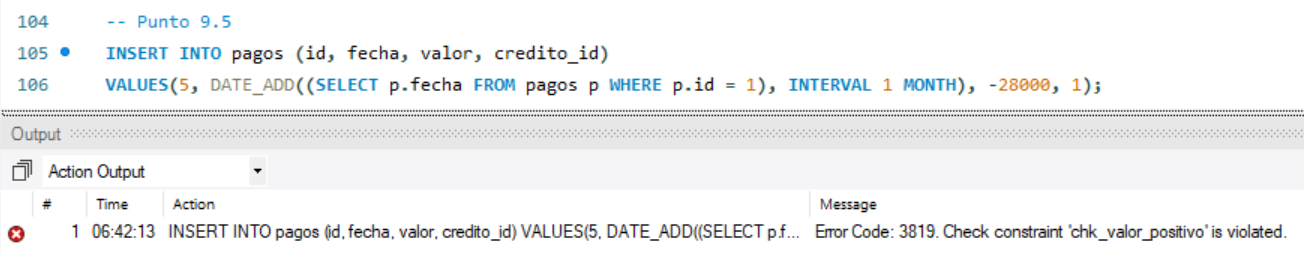


* + 1. id=5, fecha=**un mes después del primer pago,** valor=**-**28000, credito\_id=1

Sentencia SQL

INSERT INTO pagos (id, fecha, valor, credito\_id)

VALUES(5, DATE\_ADD((SELECT p.fecha FROM pagos p WHERE p.id = 1), INTERVAL 1 MONTH), -28000, 1);



Se observa en este punto que al intentar introducir un valor negativo en el pago se viola la restricción hecha en esa columna donde no se permiten valores negativos.

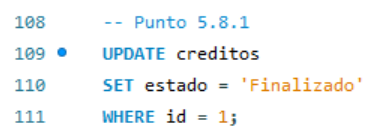
* 1. Cambiar o actualizar el estado del Crédito con id=1, para que ahora sea Finalizado.

Sentencia SQL

UPDATE creditos

SET estado = 'Finalizado'

WHERE id = 1;

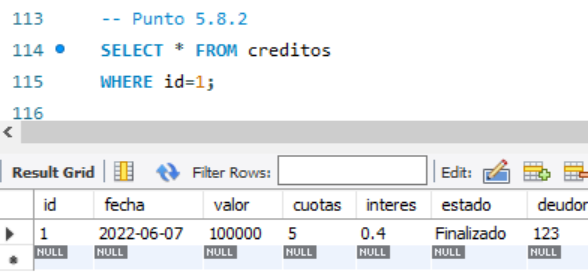


* 1. Mostrar los datos del Crédito con id=1

Sentencia SQL

SELECT \* FROM creditos

WHERE id=1;

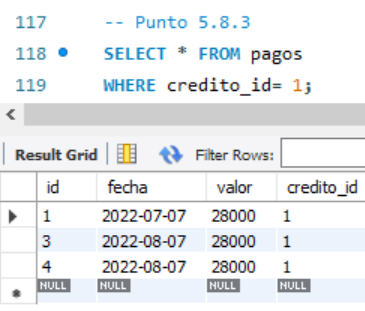


* 1. Mostrar todos los Pagos insertados al Crédito con ID=1 insertados en la BD

Sentencia SQL

SELECT \* FROM pagos

WHERE credito\_id= 1;



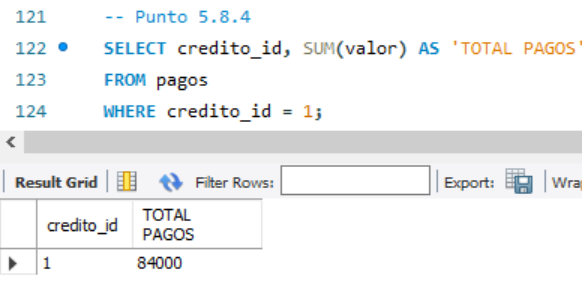
* 1. Totalizar el total de valores pagados para el crédito con id=1, para que aparezca así: TOTAL PAGOS: $140000

Sentencia SQL

SELECT credito\_id, SUM(valor) AS 'TOTAL PAGOS'

FROM pagos

WHERE credito\_id = 1;



* 1. Analice el resultado de cada una de las operaciones anteriores, comente y explique el por que de cada uno.

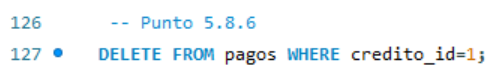
Se observa que en el crédito con id = 1 el cual corresponde a un valor de $ 100.000 con una tasa de interés del 40% mensual y cuotas a pagar 5 por una valor de $28.000 cada una. Al realizar los 5 pagos se observó que el pago con id = 2 no fue posible porque violaba la restricción hecha de que no se puede ingresar pagos con fecha posterior a la que se hace el mismo, el pago con id = 5 tampoco pudo hacerse porque viola la restricción del valor negativo.

Por tales razones al totalizar los pagos se observa que solo fueron pagados 3 cuotas y debido a eso el TOTAL PAGOS fue de $ 84.000 lo que indica que aun hay dos cuotas pendientes por pagar y el crédito no puede estar en estado = ‘Finalizado’.

* 1. Elimine todos los pagos de crédito con ID: 1

Sentencia SQL

DELETE FROM pagos WHERE credito\_id=1;

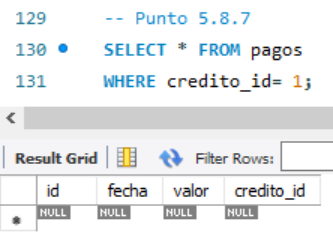


* 1. Mostrar todos los pagos que existen para el crédito con id=1

Sentencia SQL

SELECT \* FROM pagos

WHERE credito\_id= 1;



* 1. inicie una TRANSACCION
  2. dentro de la transacción volver a insertar los mismo pagos anteriores:
     1. id=1, fecha=**un mes después del crédito**, valor=28000, credito\_id=1
     2. id=2, fecha=**un mes después de la fecha actual,** valor=28000, credito\_id=1
     3. id=3, fecha=**un mes después del primer pago,** valor=28000, credito\_id=1
     4. id=4, fecha=**un mes después del primer pago,** valor=28000, credito\_id=1
     5. id=5, fecha=**un mes después del primer pago,** valor=**-**28000, credito\_id=1

Sentencia SQL

START TRANSACTION;

-- Punto 8.9.1

INSERT INTO pagos (id, fecha, valor, credito\_id)

VALUES(1, DATE\_ADD((SELECT fecha FROM creditos WHERE id = 1), INTERVAL 1 MONTH), 28000, 1);

-- Punto 8.9.2

INSERT INTO pagos (id, fecha, valor, credito\_id)

VALUES(2, DATE\_ADD(CURDATE(), INTERVAL 1 MONTH), 28000, 1);

-- Punto 8.9.3

INSERT INTO pagos (id, fecha, valor, credito\_id)

VALUES(3, DATE\_ADD((SELECT p.fecha FROM pagos p WHERE p.id = 1), INTERVAL 1 MONTH), 28000, 1);

-- Punto 8.9.4

INSERT INTO pagos (id, fecha, valor, credito\_id)

VALUES(4, DATE\_ADD((SELECT p.fecha FROM pagos p WHERE p.id = 1), INTERVAL 1 MONTH), 28000, 1);

-- Punto 8.9.5

INSERT INTO pagos (id, fecha, valor, credito\_id)

VALUES(5, DATE\_ADD((SELECT p.fecha FROM pagos p WHERE p.id = 1), INTERVAL 1 MONTH), -28000, 1);

ROLLBACK;

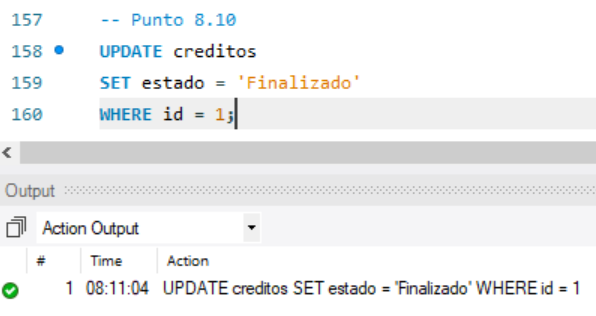
* 1. Cambiar o actualizar el estado del Crédito con id=1, para que ahora sea Finalizado

Sentencia SQL

UPDATE creditos

SET estado = 'Finalizado'

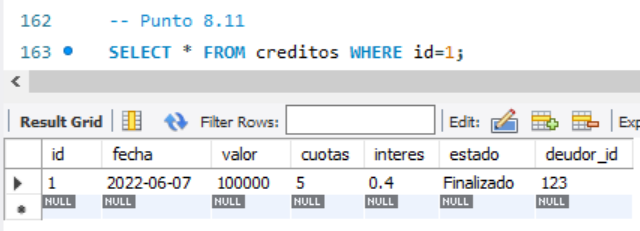
WHERE id = 1;



* 1. Mostrar los datos del Crédito con id=1

Sentencia SQL

SELECT \* FROM creditos WHERE id= 1;

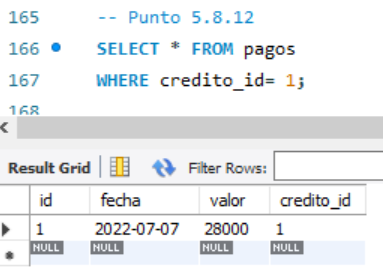


* 1. Mostrar todos los Pagos insertados al Crédito con ID=1 insertados en la BD

Sentencia SQL

SELECT \* FROM pagos

WHERE credito\_id= 1;



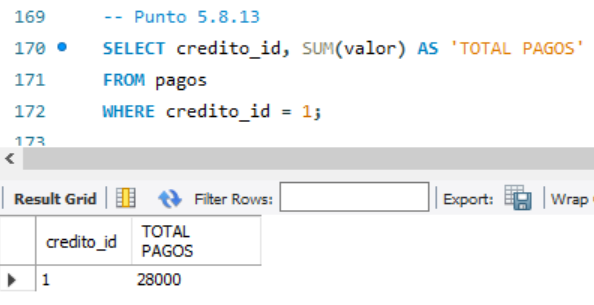
* 1. Totalizar el total de valores pagados para el crédito con id=1, para que aparezca así: TOTAL PAGOS: $140000

Sentencia SQL

SELECT credito\_id, SUM(valor) AS 'TOTAL PAGOS'

FROM pagos

WHERE credito\_id = 1;



* 1. Cancelar la transaccion (rollblack)

Sentencia SQL

ROLLBACK;

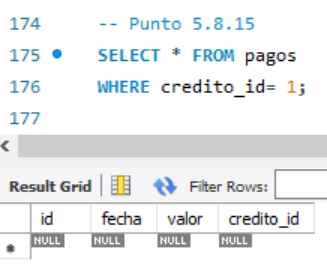


* 1. Mostrar todos los Pagos insertados al Crédito con ID=1 insertados en la BD

Sentencia SQL

SELECT \* FROM pagos

WHERE credito\_id= 1;



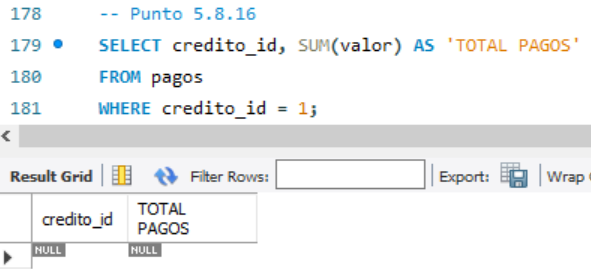
* 1. Totalizar el total de valores pagados para el crédito con id=1, para que aparezca así: TOTAL PAGOS: $140000

Sentencia SQL

SELECT credito\_id, SUM(valor) AS 'TOTAL PAGOS'

FROM pagos

WHERE credito\_id = 1;



* 1. Analice el resultado de cada una de las operaciones anteriores, comente y explique el por que de cada uno.

En este punto se observa que al hacer los pagos en la transacciones un solo pago es el que logra ingresar debido a que el segundo pago genera un error por violar la restricción de fecha, de ahí en adelante no ingresan los demás pagos, lo cual se comprueba viendo que entro un solo pago en la tabla pagos. La sentencia ROLLBACK revierte las operaciones que se encuentran dentro de la transacción, por ese motivo al ejecutar esta instrucción se observa que la tabla pagos no hay pagos.

1. Diligenciar la siguiente tabla con el mismo ejemplo por línea para los motor de bd:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EJEMPLO DE | ORACLE | POSTGRESQL | SQL SERVER | MYSQL |
| Subconsulta en el Select |  |  |  |  |
| Subconsulta en el From |  |  |  |  |
| Subconsulta en el Where |  |  |  |  |
| Subconsulta en el Having |  |  |  |  |
| Uso de Check |  |  |  |  |
| Join Inner |  |  |  |  |
| Join Full |  |  |  |  |
| Join Outter |  |  |  |  |
| Join Lef |  |  |  |  |
| Join Rigth |  |  |  |  |
| Unión |  |  |  |  |
| Intersect |  |  |  |  |
| Crear una Vista SQL |  |  |  |  |
| Usar una Vista SQL |  |  |  |  |
| Editar una Vista SQL |  |  |  |  |
| Mostrar el contenido de una Vista SQL |  |  |  |  |
| Mostrar las Vistas SQL |  |  |  |  |
| Eliminar una Vista SQL |  |  |  |  |
| Crear una Transacción SQL |  |  |  |  |
| Uso de Try para detectar cuando falla una transacción SQL |  |  |  |  |

## MySQL

* + 1. Ejemplo de subconsulta en el Select.

Cada miembro del grupo debe realizar una video explicando este cuadro en MySQL y en el motor de bd seleccionado. Es necesario utilizar cámara y voz propia.

El video debe ser subido a youtube (ocultarlo si lo desea) y compartir el enlace al correo institucional (jarrietaa2@unicartagena.edu.co)