**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 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

Explicación: Una vista en SQL es como una ventana virtual a una consulta. No almacena datos por sí sola, sino que muestra resultados de una consulta SELECT como si fuera una tabla.

¿Qué as͏unto ͏soluciona? **Facili͏ta preguntas complicadas:** una ͏pregunta grande se p͏uede guardar como vista para no volver a hacerla.

**Seg͏ur͏idad y c͏ontrol de en͏trada:** se p͏uede enseñar solo parte de los dato͏s (p͏or ej͏emplo, no mostrar salarios).

**Ab͏stracción d͏e información:** los ͏us͏uarios ve͏n ͏la vi͏sta co͏mo una ͏tabla; no tienen que conocer las unen, filtro͏s, o cuentas detrás.

1. 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).

Instrucción SQL:

**SELECT**

**CONCAT(c.Nombres, ' ', c.Apellidos) AS Nombre\_Cliente,**

**a.Marca AS Marca\_Auto,**

**a.Tipo AS Tipo\_Auto,**

**a.Color AS Color\_Auto,**

**ov.Fecha\_Solicitud,**

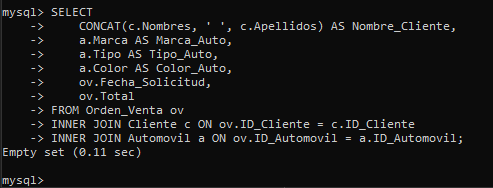
**ov.Total**

**FROM Orden\_Venta ov**

**INNER JOIN Cliente c ON ov.ID\_Cliente = c.ID\_Cliente**

**INNER JOIN Automovil a ON ov.ID\_Automovil = a.ID\_Automovil;**

Captura de pantalla:



1. Usar la consulta anterior y crear con ella una Vista SQL

Instrucción SQL:

**CREATE VIEW Vista\_Ordenes\_Clientes AS**

**SELECT**

**CONCAT(c.Nombres, ' ', c.Apellidos) AS Nombre\_Cliente,**

**a.Marca AS Marca\_Auto,**

**a.Tipo AS Tipo\_Auto,**

**a.Color AS Color\_Auto,**

**ov.Fecha\_Solicitud,**

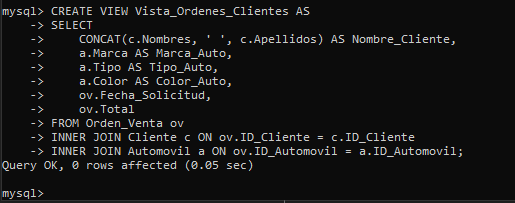
**ov.Total**

**FROM Orden\_Venta ov**

**INNER JOIN Cliente c ON ov.ID\_Cliente = c.ID\_Cliente**

**INNER JOIN Automovil a ON ov.ID\_Automovil = a.ID\_Automovil;**

Captura de pantalla:

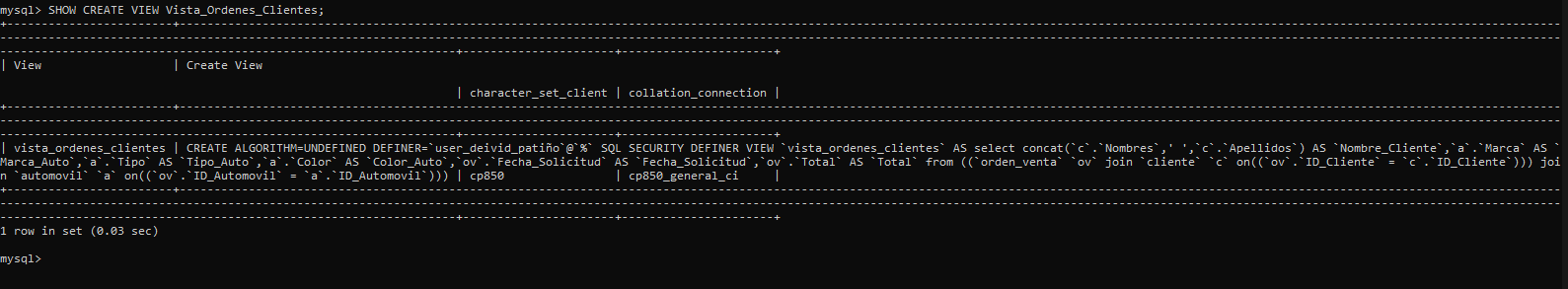


1. Mostrar la Vista SQL creada anteriormente

Instrucción SQL:

**SHOW CREATE VIEW Vista\_Ordenes\_Clientes;**

Captura de pantalla:

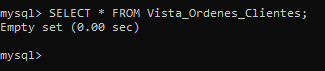


1. Ejecutar la Vista SQL creada anteriormente

Instrucción SQL:

**SELECT \* FROM Vista\_Ordenes\_Clientes;**

Captura de pantalla:



1. Abrir la Vista SQL y modificarla para cambiar la consulta por otra consulta o modificar la existente.

Instrucción SQL:

**DROP VIEW IF EXISTS Vista\_Ordenes\_Clientes;**

**CREATE VIEW Vista\_Ordenes\_Clientes AS**

**SELECT**

**CONCAT(c.Nombres, ' ', c.Apellidos) AS Nombre\_Cliente,**

**a.Marca AS Marca\_Auto,**

**a.Tipo AS Tipo\_Auto,**

**a.Color AS Color\_Auto,**

**ov.Fecha\_Solicitud,**

**ov.Total,**

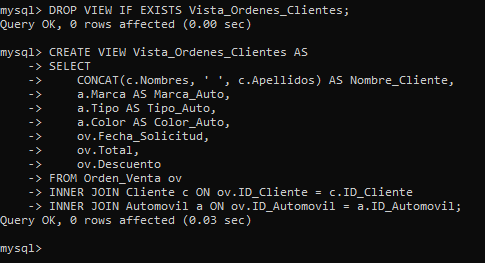
**ov.Descuento**

**FROM Orden\_Venta ov**

**INNER JOIN Cliente c ON ov.ID\_Cliente = c.ID\_Cliente**

**INNER JOIN Automovil a ON ov.ID\_Automovil = a.ID\_Automovil;**

Captura de pantalla:

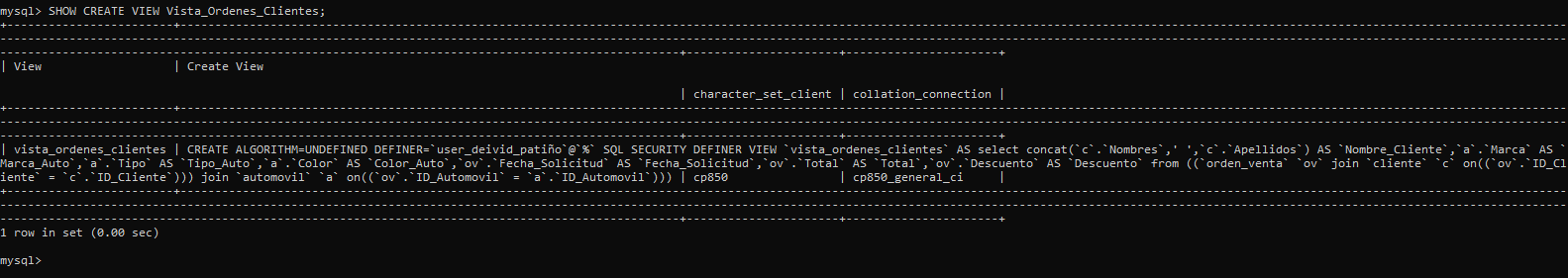


1. Volver a ejecutar la Vista SQL

Instrucción SQL:

**SHOW CREATE VIEW Vista\_Ordenes\_Clientes;**

Captura de pantalla:



1. 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

Explicación: Las vistas permiten organizar mejor las consultas complejas y reutilizarlas fácilmente también ayudan a mejorar la seguridad ya que se pueden ocultar columnas sensibles y mostrar solo la información necesaria además, se pueden usar como si fueran tablas normales en otras consultas, lo que facilita el trabajo.

* 1. Describir las desventajas del uso de Vistas SQL para el X motor de bases de datos sobre el cual desarrolló la actividad

Explicación: MySQL no tiene buen soporte para vistas actualizables

especialmente si contienen JOINS En algunos casos no se pueden usar para insertar o actualizar datos lo que limita su utilidad y si se usan demasiado en consultas grandes pueden afectar el rendimiento.

* 1. 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.

Explicación: Hay cosas que las vistas no permiten como usar ORDER BY directamente en su definición también pueden fallar con funciones como RAND o NOW. Si la vista es muy compleja no siempre se puede actualizar o insertar datos desde ella.

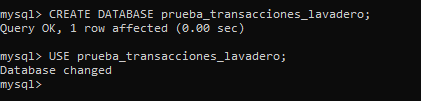
1. Para el desarrollo de los siguientes puntos referentes al tema de **Transacciones**, ustedes como grupo deben realizar las siguientes tareas sobre OracleSql, SQLServer y PostgreSQL y MySQL:
   1. Crear una base de datos llamada prueba\_transacciones\_grupoXYZ, siendo XYZ el nombre del ejercicio escogido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

Instrucción SQL:

**CREATE DATABASE prueba\_transacciones\_lavadero;**

**USE prueba\_transacciones\_lavadero;**

Captura de pantalla:



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

Instrucción SQL:

**CREATE TABLE Deudores (**

**cc VARCHAR(20) PRIMARY KEY,**

**clave VARCHAR(50) NOT NULL,**

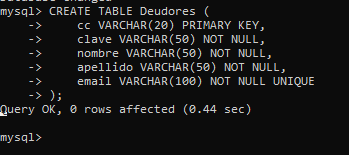
**nombre VARCHAR(50) NOT NULL,**

**apellido VARCHAR(50) NOT NULL,**

**email VARCHAR(100) NOT NULL UNIQUE**

**);**

Captura de pantalla:



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

Instrucción SQL:

**CREATE TABLE Creditos (**

**id INT PRIMARY KEY,**

**fecha DATE NOT NULL,**

**valor DECIMAL(10,2) NOT NULL,**

**cuotas INT NOT NULL,**

**interes DECIMAL(5,2) NOT NULL,**

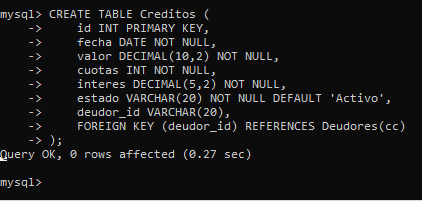
**estado VARCHAR(20) NOT NULL DEFAULT 'Activo',**

**deudor\_id VARCHAR(20),**

**FOREIGN KEY (deudor\_id) REFERENCES Deudores(cc)**

**);**

Captura de pantalla:



* 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 mauyor que la fecha actual

Instrucción SQL:

**CREATE TABLE Pagos (**

**id INT PRIMARY KEY,**

**fecha DATE NOT NULL CHECK (fecha <= CURDATE()),**

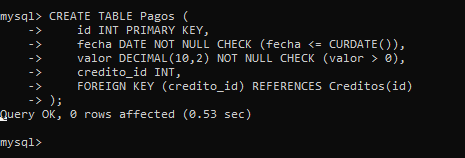
**valor DECIMAL(10,2) NOT NULL CHECK (valor > 0),**

**credito\_id INT,**

**FOREIGN KEY (credito\_id) REFERENCES Creditos(id)**

**);**

Captura de pantalla:



* 1. Insertar un al menor un Deudor:

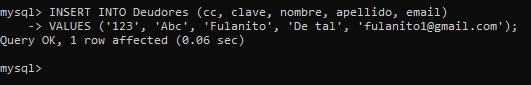
El primero con cc=123, calve=Abc, nombre = Fulanito, apellido=De tal, email=[fulanito1@gmail.com](mailto:fulanito1@gmail.com)}

Instrucción SQL:

**INSERT INTO Deudores (cc, clave, nombre, apellido, email)**

**VALUES ('123', 'Abc', 'Fulanito', 'De tal', 'fulanito1@gmail.com');**

Captura de pantalla:

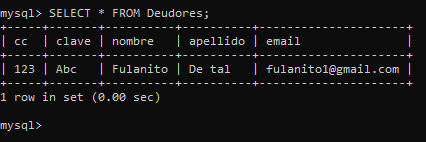


* 1. Mostrar todos los Deudores insertados en la BD

Instrucción SQL:

**SELECT \* FROM Deudores;**

Captura de pantalla:



* 1. Insertar al menos un Crédito:

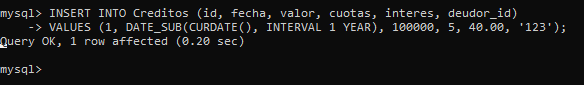
id=1, fecha=hace un año, valor=100000, cuotas=5, interes\_mes=40%, deudor\_id=123

Instrucción SQL:

**INSERT INTO Creditos (id, fecha, valor, cuotas, interes, deudor\_id)**

**VALUES (1, DATE\_SUB(CURDATE(), INTERVAL 1 YEAR), 100000, 5, 40.00, '123');**

Captura de pantalla:

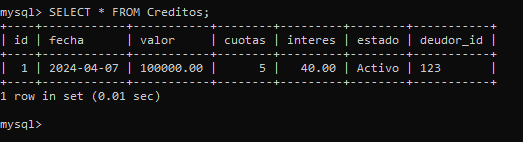


* 1. Mostrar todos los Créditos insertados en la BD

Instrucción SQL:

**SELECT \* FROM Creditos;**

Captura de pantalla:

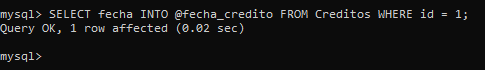


* 1. Insertar 5 pagos así (no modifique ningún valor del ejemplo, colocar las fechas correspondientes, el formato de fecha correspondiente para bd):

Instrucción SQL:

**SELECT fecha INTO @fecha\_credito FROM Creditos WHERE id = 1;**

Captura de pantalla:

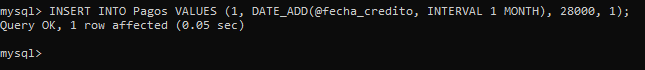


* + 1. id=1, fecha=**un mes después del crédito**, valor=28000, credito\_id=1

Instrucción SQL:

**INSERT INTO Pagos VALUES (1, DATE\_ADD(@fecha\_credito, INTERVAL 1 MONTH), 28000, 1);**

Captura de pantalla:

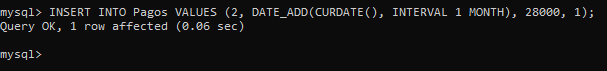


* + 1. id=2, fecha=**un mes después de la fecha actual,** valor=28000, credito\_id=1

Instrucción SQL:

**INSERT INTO Pagos VALUES (2, DATE\_ADD(CURDATE(), INTERVAL 1 MONTH), 28000, 1);**

Captura de pantalla:



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

Instrucción SQL:

**INSERT INTO Pagos VALUES (3, DATE\_ADD(@fecha\_credito, INTERVAL 2 MONTH), 28000, 1);**

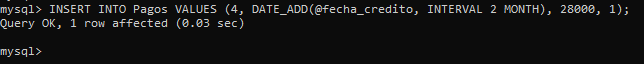
Captura de pantalla:



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

Instrucción SQL:  
**INSERT INTO Pagos VALUES (4, DATE\_ADD(@fecha\_credito, INTERVAL 2 MONTH), 28000, 1);**

Captura de pantalla:

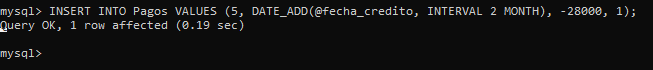


* + 1. id=5, fecha=**un mes después del primer pago,** valor=**-**28000, credito\_id=1

Instrucción SQL:

**INSERT INTO Pagos VALUES (5, DATE\_ADD(@fecha\_credito, INTERVAL 2 MONTH), -28000, 1);**

Captura de pantalla:



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

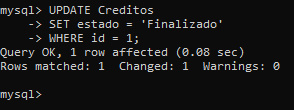
Instrucción SQL:

**UPDATE Creditos**

**SET estado = 'Finalizado'**

**WHERE id = 1;**

Captura de pantalla:



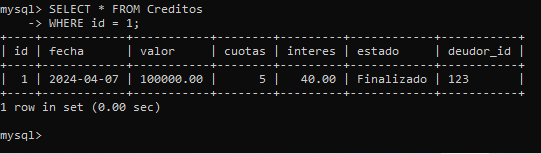
* 1. Mostrar los datos del Crédito con id=1

Instrucción SQL:

**SELECT \* FROM Creditos**

**WHERE id = 1;**

Captura de pantalla:



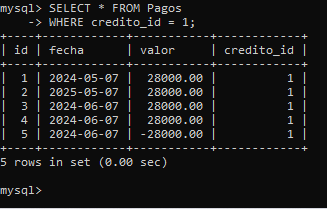
* 1. Mostrar todos los Pagos insertados al Crédito con ID=1 insertados en la BD

Instrucción SQL:

**SELECT \* FROM Pagos**

**WHERE credito\_id = 1;**

Captura de pantalla:



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

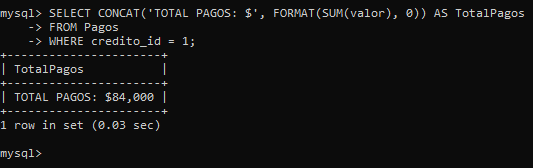
Instrucción SQL:

**SELECT CONCAT('TOTAL PAGOS: $', FORMAT(SUM(valor), 0)) AS TotalPagos**

**FROM Pagos**

**WHERE credito\_id = 1;**

Captura de pantalla:



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

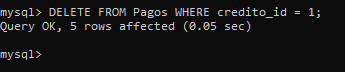
Explicación: El pago 2 no se agregó porque tení͏a una fecha futura, lo que incumple l͏a regla CHECK fecha <= CURDATE() el pago 5 falló porque el valor era negativo, y eso rompe el CHECK valor > ͏0 el estado del pago͏ se cambió bien usando un UPDATE.

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

Instrucción SQL:

**DELETE FROM Pagos WHERE credito\_id = 1;**

Captura de pantalla:

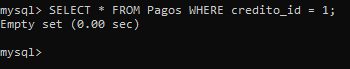


* 1. Mostrar todos los pagos que existen para el crédito con id=1

Instrucción SQL:

**SELECT \* FROM Pagos WHERE credito\_id = 1;**

Captura de pantalla:

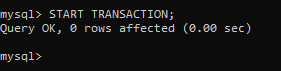


* 1. inicie una TRANSACCION

Instrucción SQL:

**START TRANSACTION;**

Captura de pantalla:

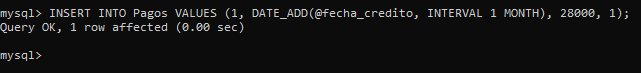


* 1. dentro de la transacción volver a insertar los mismo pagos anteriores:
     1. id=1, fecha=**un mes después del crédito**, valor=28000, credito\_id=1

Instrucción SQL:

**INSERT INTO Pagos VALUES (1, DATE\_ADD(@fecha\_credito, INTERVAL 1 MONTH), 28000, 1);**

Captura de pantalla:

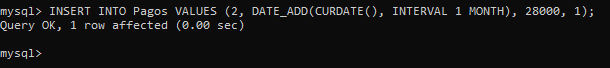


* + 1. id=2, fecha=**un mes después de la fecha actual,** valor=28000, credito\_id=1

Instrucción SQL:

**INSERT INTO Pagos VALUES (2, DATE\_ADD(CURDATE(), INTERVAL 1 MONTH), 28000, 1);**

Captura de pantalla:

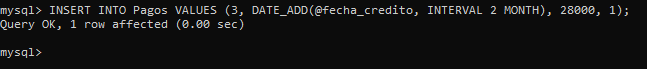


* + 1. id=3, fecha=**un mes después del primer pago,** valor=28000, credito\_id=1

Instrucción SQL:

**INSERT INTO Pagos VALUES (3, DATE\_ADD(@fecha\_credito, INTERVAL 2 MONTH), 28000, 1);**

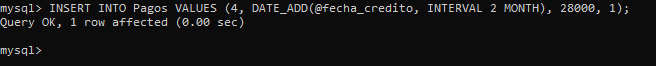
Captura de pantalla:



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

Instrucción SQL:  
**INSERT INTO Pagos VALUES (4, DATE\_ADD(@fecha\_credito, INTERVAL 2 MONTH), 28000, 1);**

Captura de pantalla:

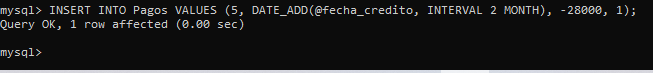


* + 1. id=5, fecha=**un mes después del primer pago,** valor=**-**28000, credito\_id=1

Instrucción SQL:

**INSERT INTO Pagos VALUES (5, DATE\_ADD(@fecha\_credito, INTERVAL 2 MONTH), -28000, 1);**

Captura de pantalla:

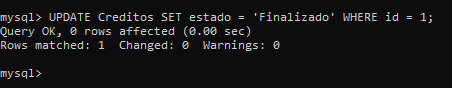


* 1. Cambiar o actualizar el estado del Crédito con id=1, para que ahora sea Finalizado

Instrucción SQL:

**UPDATE Creditos SET estado = 'Finalizado' WHERE id = 1;**

Captura de pantalla:

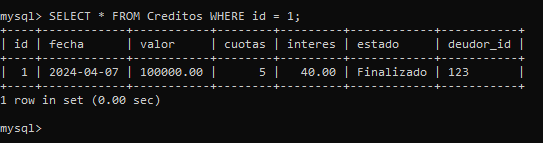


* 1. Mostrar los datos del Crédito con id=1

Instrucción SQL:

**SELECT \* FROM Creditos WHERE id = 1;**

Captura de pantalla:

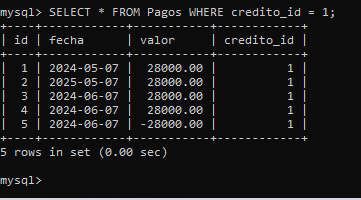


* 1. Mostrar todos los Pagos insertados al Crédito con ID=1 insertados en la BD

Instrucción SQL:

**SELECT \* FROM Pagos WHERE credito\_id = 1;**

Captura de pantalla:



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

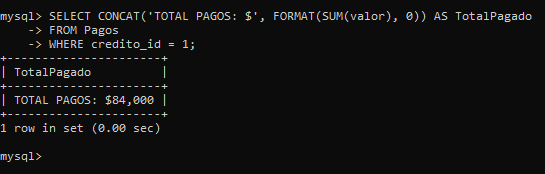
Instrucción SQL:

**SELECT CONCAT('TOTAL PAGOS: $', FORMAT(SUM(valor), 0)) AS TotalPagado**

**FROM Pagos**

**WHERE credito\_id = 1;**

Captura de pantalla:

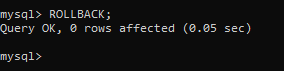


* 1. Cancelar la transaccion (rollblack)

Instrucción SQL:

**ROLLBACK;**

Captura de pantalla:

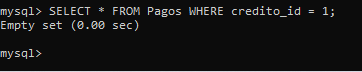


* 1. Mostrar todos los Pagos insertados al Crédito con ID=1 insertados en la BD

Instrucción SQL:

**SELECT \* FROM Pagos WHERE credito\_id = 1;**

Captura de pantalla:



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

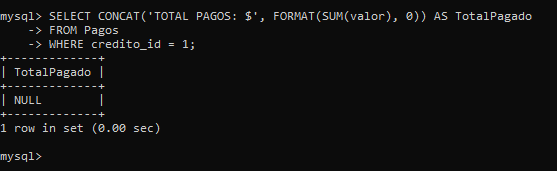
Instrucción SQL:

**SELECT CONCAT('TOTAL PAGOS: $', FORMAT(SUM(valor), 0)) AS TotalPagado**

**FROM Pagos**

**WHERE credito\_id = 1;**

Captura de pantalla:



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

Explicacion: Un ROLLBACK hace que un cambio o inserción incorrecta se cancelen. Esto también se detiene͏ cualq͏ui͏er ͏otro cambi͏o que͏ se estuviera haciendo también se ve el poder del CHECK no dejan poner datos incorrectos, como pagos malos o fechas venideras.