importar sqlite3

de error de importación de sqlite3

importar fecha y hora

sistema de importación

importar sistema operativo

de la importación de archivos tarExtractError

importar openpyxl

def verificarId ( id , tabla ):

con sqlite3 . connect ( "COWORKING.db" ) como conexión :

mi\_cursor = conexión . cursor ()

mi\_cursor . ejecutar ( "SELECCIONAR \* DESDE" + tabla + "DONDE id" + tabla + "=" + str ( id ))

valor = mi\_cursor . buscar ()

si no valor :

volver falso

más :

volver verdadero

#Esta funcion se usara para todo lo que tenga un ingreso de informacion como INSERT UPDATE DELETE

def IngresoBD ( comando , valores ):

prueba :

con sqlite3 . connect ( "COWORKING.db" ) como conexión :

mi\_cursor = conexión . cursor ()

mi\_cursor . ejecutar ( comando , valores )

excepto Error como e :

imprimir ( e )

volver falso

excepto :

print ( f"Se produjo el siguiente error: { sys . exc\_info ()[ 0 ] } " )

volver falso

finalmente :

contacto \_ cerrar ()

volver verdadero

#Esta funcion se usa para cualquier tipo de extraccion de informacion de la BD como SELECT

def ExtracciónBD ( comando , valores ):

prueba :

con sqlite3 . connect ( "COWORKING.db" ) como conexión :

mi\_cursor = conexión . cursor ()

mi\_cursor . ejecutar ( comando , valores )

devuelve mi\_cursor . buscar ()

excepto Error como e :

imprimir ( e )

excepto :

print ( f"Se produjo el siguiente error: { sys . exc\_info ()[ 0 ] } " )

finalmente :

contacto \_ cerrar ()

# Generamos la base de datos y las tablas para el script

prueba :

con sqlite3 . connect ( "COWORKING.db" ) como conexión :

mi\_cursor = conexión . cursor ()

mi\_cursor . ejecutar ( """CREAR TABLA SI NO EXISTE Sala (

idSala CLAVE PRIMARIA ENTERA,

nombreSala TEXTO NO NULO,

cupoSala INTEGER);""" )

mi\_cursor . ejecutar ( """CREAR TABLA SI NO EXISTE Cliente (

idCliente CLAVE PRIMARIA ENTERA,

nombreCliente TEXTO NO NULO);""" )

mi\_cursor . ejecutar ( """CREAR TABLA SI NO EXISTE Turno (

idTurno CLAVE PRIMARIA ENTERA,

nombreTurno TEXTO NO NULO);""" )

mi\_cursor . ejecutar ( """CREAR TABLA SI NO EXISTE Reserva (

idReserva CLAVE PRIMARIA ENTERA,

nombreReserva TEXTO NO NULO,

fechaReserva TEXTO NO NULO,

cliente ENTERO NO NULO,

sala ENTERO NO NULO,

a su vez ENTERO NO NULO,

CLAVE EXTRANJERA(cliente) REFERENCIAS sala(idCliente),

CLAVE EXTRANJERA(sala) REFERENCIAS cliente(idSala),

CLAVE EXTRANJERA(turno) REFERENCIAS turno(idTurno));""" )

aprobar

excepto Error como e :

imprimir ( e )

# Agregamos los turnos a la tabla "turno".

prueba :

con sqlite3 . connect ( "COWORKING.db" ) como conexión :

mi\_cursor = conexión . cursor ()

mi\_cursor . ejecutar ( "INSERTAR EN VALORES de turno (1, 'Matutino')" )

mi\_cursor . ejecutar ( "INSERTAR EN VALORES de turno (2, 'Vespertino')" )

mi\_cursor . ejecutar ( "INSERTAR EN VALORES de turno (3, 'Nocturno')" )

excepto Error como e :

imprimir ( e )

excepto :

print ( f"Se produjo el siguiente error: { sys . exc\_info ()[ 0 ] } " )

finalmente :

contacto \_ cerrar ()

# ---- MENÚ PRINCIPAL ----

os \_ sistema ( "cls" )

mientras que es cierto :

os \_ sistema ( "cls" )

imprimir ( "-" \* 50 )

imprimir ( "MENÚ PRINCIPAL" )

print ( " \t [A] Reservaciones." )

print ( " \t [B] Informes." )

print ( " \t [C] Registradora Nueva Sala." )

print ( " \t [D] Registrador Nuevo Cliente." )

imprimir ( " \t [X] Salir." )

op\_principal = entrada ( "OPCION:" )

os \_ sistema ( "cls" )

# OPCION A. RESERVAS

si ( op\_principal . superior () == "A" ):

mientras que es cierto :

# Recolección de todos los registros en la BD

valores = {}

registrosClientes = ExtraccionBD ( "SELECCIONAR \* DESDE Cliente ORDENAR POR idCliente" , valores )

registrosSalas = ExtraccionBD ( "SELECCIONAR \* DESDE Sala ORDENAR POR idSala" , valores )

registrosTurnos = ExtraccionBD ( "SELECCIONAR \* DESDE Turno ORDENAR POR idTurno" , valores )

registrosReservas = ExtraccionBD ( "SELECCIONE \* DESDE Reserva ORDEN POR idReserva" , valores )

r = {}

registrosClientes = ExtraccionBD ( "SELECCIONAR \* DESDE Cliente ORDENAR POR idCliente" , r )

registrosSalas = ExtraccionBD ( "SELECCIONAR \* DESDE Sala ORDENAR POR idSala" , r )

registrosTurnos = ExtraccionBD ( "SELECCIONE \* DESDE Turno ORDENAR POR idTurno" , r )

registrosReservas = ExtraccionBD ( "SELECCIONAR \* DESDE Reserva ORDENAR POR idReserva" , r )

# Submenú Reservaciones

os \_ sistema ( "cls" )

@@ -138,9 +138,7 @@ def ExtraccionBD(comando,valores):

#Impresion de cliente

imprimir ( "-" \* 54 )

print ( f'| { "RESERVACION DE SALA" :^52 } |' )

imprimir ( "-" \* 54 )

imprimir ( f'| { "Clientes" :^52 } |' )

imprimir ( f'| { "CLIENTES" :^52 } |' )

imprimir ( "-" \* 54 )

print ( f'| { " Numero Cliente" :^15 } { "Nombre" :^37 } |' )

imprimir ( "-" \* 54 )

@@ -166,12 +164,12 @@ def ExtraccionBD(comando,valores):

os \_ sistema ( "cls" )

imprimir ( "-" \* 54 )

print ( f'| { " RESERVACION DE SALA " :^52 } |' )

print ( f'| { " FECHA DEL EVENTO " :^52 } |' )

imprimir ( "-" \* 54 )

# Validacion de fecha

mientras que es cierto :

fecha\_str = input ( " \ n Fecha del evento dd/mm/aaaa: " )

fecha\_str = input ( " \ n Ingrese la fecha deseada dd/mm/aaaa: " )

prueba :

fechaReserva = fechahora . fecha y hora strptime ( fecha\_str , "%d/%m/%Y" ). fecha ()

fecha\_actual = fechahora . fecha \_ hoy ()

@@ -188,10 +186,8 @@ def ExtraccionBD(comando,valores):

# Impresion de sala

imprimir ( "-" \* 54 )

print ( f'| { " RESERVACION DE SALA " :^52 } |' )

imprimir ( f'| { " SALAS " :^52 } |' )

imprimir ( "-" \* 54 )

imprimir ( f'| { "Salas" :^52 } |' )

imprimir ( "-" \* 54 )

print ( f'| { " Numero Sala" :^15 } { "Nombre" :^27 } { "Capacidad " :^10 } |' )

imprimir ( "-" \* 54 )

para idSala , nombreSala , cupoSala en registrosSalas :

@@ -217,10 +213,8 ​​@@ def ExtraccionBD(comando,valores):

# Impresion de turno

imprimir ( "-" \* 54 )

print ( f'| { " RESERVACION DE SALA " :^52 } |' )

print ( f'| { " TURNOS " :^52 } |' )

imprimir ( "-" \* 54 )

print ( f'| { "Turnos" :^52 } |' )

imprimir ( "-" \* 54 )

print ( f'| { " Numero Turno" :^15 } { "Turno" :^37 } |' )

imprimir ( "-" \* 54 )

para idTurno , nombreTurno en registrosTurnos :

@@ -248,12 +242,12 @@ def ExtraccionBD(comando,valores):

valores = { "fechaReserva" : fechaReserva , "idSala" : sala , "idTurno" : turno }

si no ExtraccionBD ( "SELECCIONA \*DESDE Reserva DONDE fechaReserva=:fechaReserva y sala=:idSala y turno=:idTurno" , valores ):

imprimir ( "-" \* 54 )

print ( f'| { " RESERVACION DE SALA " :^52 } |' )

print ( f'| { " NOMBRE DE LA RESERVACION " :^52 } |' )

imprimir ( "-" \* 54 )

# Validación de nombre

mientras que es cierto :

nombreReserva = input ( ' \ n Nombre para la reserva : ' )

if nombreReserva == "" :

nombreReserva = input ( ' \ n Ingrese el nombre para la reserva : ' )

if nombreReserva == "" o nombreReserva . isspace () :

print ( "No se puede omitir." )

continuar

más :

@@ -306,7 +300,7 @@ def ExtraccionBD(comando,valores):

# Validación del nuevo nombre para la reserva

mientras que es cierto :

nuevoNombre = input ( 'Nuevo nombre para el evento: ' )

si nuevoNombre == "" :

si nuevoNombre == "" o nuevoNombre . isspace () :

print ( 'Nombre no valido!' )

más :

romper

@@ -449,20 +443,6 @@ def ExtraccionBD(comando,valores):

# A. Reporte en Pantalla

if ( op\_reporte . superior () == "A" ):

# Extraccion de informacion de reportes de la BD

valores = {}

todasReservas = ExtraccionBD ( "SELECT Reserva.sala, Cliente.nombreCliente, Reserva.nombreReserva, Turno.nombreTurno, Reserva.fechaReserva FROM Reserva INNER JOIN Cliente ON Reserva.cliente = Cliente.idCliente INNER JOIN Turno on Reserva.turno = Turno.idTurno" , valores )

#Impresión de Reporte

imprimir ( "\*" \* 85 )

print ( f'\*\* { "REPORTE DE RESERVAS " :^81 } \*\*' )

imprimir ( "\*" \* 85 )

print ( f' { "SALA" :<10 } { "CLIENTE" :<20 } { "EVENTO" :<20 } { "TURNO" :<20 } { "FECHA" :<26 } ' )

imprimir ( '\*' \* 85 )

para idSala , nombreCliente , nombreReserva , Turno , fechaReserva en todasReservas :

print ( f' { idSala :<10 } { nombreCliente :<20 } { nombreReserva :<20 } { Turno :<20 } { fechaReserva :<26 } ' )

imprimir ( f' { "FIN DEL REPORTE" :\*^85 } ' )

# Validacion de fecha

mientras que es cierto :

fecha\_str = input ( " \n Ingrese la fecha del evento a buscar dd/mm/aaaa: " )

@@ -476,34 +456,45 @@ def ExtraccionBD(comando,valores):

# Extracción de información de reportes de la BD con fecha especifica

valores = { "fechaReserva" : fechaReserva }

registrosReservasFecha = ExtraccionBD ( "SELECT Reserva.sala , Cliente.nombreCliente, Reserva.nombreReserva, Turno.nombreTurno FROM Reserva INNER JOIN Cliente ON Reserva.cliente = Cliente.idCliente INNER JOIN Turno on Reserva.turno = Turno.idTurno WHERE Reserva.fechaReserva = :fechaReserva" , valores )

registrosReservasFecha = ExtraccionBD ( "SELECT Sala.nombreSala , Cliente.nombreCliente, Reserva.nombreReserva, Turno.nombreTurno FROM Reserva INNER JOIN Sala ON Reserva.sala = Sala.idSala INNER JOIN Cliente ON Reserva.cliente = Cliente.idCliente INNER JOIN Turno on Reserva .turno = Turno.idTurno DONDE Reserva.fechaReserva = :fechaReserva" , valores )

#Impresion de Reporte por fecha

imprimir ( "\*" \* 85 )

print ( f'\*\* { "REPORTE DE RESERVACIONES PARA EL DIA " + fecha\_str :^81 } \*\*' )

imprimir ( "\*" \* 85 )

print ( f' { "SALA" :< 7 } { "CLIENTE" :< 33 } { "EVENTO" :<33 } { "TURNO" :<12 } ' )

print ( f' { "SALA" :< 20 } { "CLIENTE" :< 20 } { "EVENTO" :<33 } { "TURNO" :<12 } ' )

imprimir ( '\*' \* 85 )

para idSala , nombreCliente , nombreReserva , Turno en registrosReservasFecha :

print ( f' { idSala :<7 } { nombreCliente :< 33 } { nombreReserva :<33 } { Turno :<12 } ' )

para nombreSala , nombreCliente , nombreReserva , Turno en registrosReservasFecha :

print ( f' { nombreSala :<20 } { nombreCliente :< 20 } { nombreReserva :<33 } { Turno :<12 } ' )

imprimir ( f' { "FIN DEL REPORTE" :\*^85 } ' )

input ( ' \n Presione cualquier tecla para continuar...' )

os \_ sistema ( "cls" )

# B. Reporte en Excel

elif ( op\_reporte . superior () == "B" ):

#Extraer reservaciones de la BD

valores = {}

registrosreserva = ExtraccionBD ( "SELECT Reserva.sala, Cliente.nombreCliente, Reserva.nombreReserva, Turno.nombreTurno, Reserva.fechaReserva FROM Reserva INNER JOIN Cliente ON Reserva.cliente = Cliente.idCliente INNER JOIN Turno on Reserva.turno = Turno.idTurno" , valores )

# Validacion de fecha

mientras que es cierto :

fecha\_str = input ( " \n Ingrese la fecha del evento a buscar dd/mm/aaaa: " )

prueba :

fechaReserva = fechahora . fecha y hora strptime ( fecha\_str , "%d/%m/%Y" ). fecha ()

romper

excepto ValueError :

print ( f"ERROR! Formato de fecha NO valida." )

print ( "Ingreso de nuevo..." )

os \_ sistema ( "cls" )

# Extracción de información de reportes de la BD con fecha especifica

valores = { "fechaReserva" : fechaReserva }

registrosReservas = ExtraccionBD ( "SELECT Sala.nombreSala, Cliente.nombreCliente, Reserva.nombreReserva, Turno.nombreTurno, Reserva.fechaReserva FROM Reserva INNER JOIN Sala ON Reserva.sala = Sala.idSala INNER JOIN Cliente ON Reserva.cliente = Cliente.idCliente INNER ÚNETE a Turno en Reserva.turno = Turno.idTurno DONDE Reserva.fechaReserva = :fechaReserva" , valores )

#Exportacion para un excel

libro = openpyxl . libro de trabajo ()

#libro.iso\_dates = Verdadero #Acepto fechas

hoja = libro \_ \_ \_

hoja \_ title = "Informe"

hoja \_ append (( 'SALA' , 'CLIENTE' , 'EVENTO' , 'TURNO' , 'FECHA' ))

para idSala , nombreCliente , nombreReserva , Turno , fechaReserva en registrosreserva :

informe = idSala , nombreCliente , nombreReserva , Turno , fechaReserva

para nombreSala , nombreCliente , nombreReserva , Turno , fechaReserva en registrosReservas :

informe = nombreSala , nombreCliente , nombreReserva , Turno , fechaReserva

hoja \_ adjuntar ( reportar )

libro \_ guardar ( "Informe.xlsx" )

input ( " \n Se exporto en excel las reservas exitosamente!" )

@@ -525,7 +516,7 @@ def ExtraccionBD(comando,valores):

imprimir ()

mientras que es cierto :

nombreSala = ( input ( "Nombre para la sala: " ))

if nombreSala == "" :

if nombreSala == "" o nombreSala . isspace () :

print ( "No se puede omitir." )

continuar

más :

@@ -555,7 +546,7 @@ def ExtraccionBD(comando,valores):

imprimir ()

mientras que es cierto :

nombreCliente = input ( "Nombre del cliente: " )

if nombreCliente == "" :

if nombreCliente == "" o nombreCliente . isspace () :

print ( "No se puede omitir." )

continuar

más :

romper

# Guardar en base de datos.

valores = ( nombreCliente ,)

if IngresoBD ( "INSERT INTO Cliente(nombreCliente) VALORES(?)" , valores ):

input ( " \n El cliente ha sido guardado exitosamente!" )

os \_ sistema ( "cls" )

# OPCION X Salir del menú principal.

elif ( op\_principal . superior () == "X" ):

romper

más :

input ( "OPCION NO VALIDA. Debe ingresar la letra que corresponde a la acción deseada." )

os \_ sistema ( "cls" )