



Universidad Autónoma de Querétaro

Facultad de Informática

Diplomado en Oracle 11g

Proyecto Final – Jorge Espinosa Avila



- ❖ Definición de la empresa (modelo de negocio, ver Anexo 1) (Fecha recomendada: 05/Ene/22)

La cadena de tiendas SAMS club requieren de un control digitalizado y mas sencillo para su gestión de sus tiendas.

Se requieren 7 tablas, departamentos, empleados, proveedores, sucursales, productos, ciudades y clientes. Para cada una de estas tablas hay datos en común como un ID y un nombre, en tablas mas especificas como empleados, se requiere además de los datos en común, un puesto, salario y jefe.

En sucursales también se pide una dirección y estado, mientras que en la tabla de productos también se pide un precio, finalmente en clientes se pide un correo y teléfono.

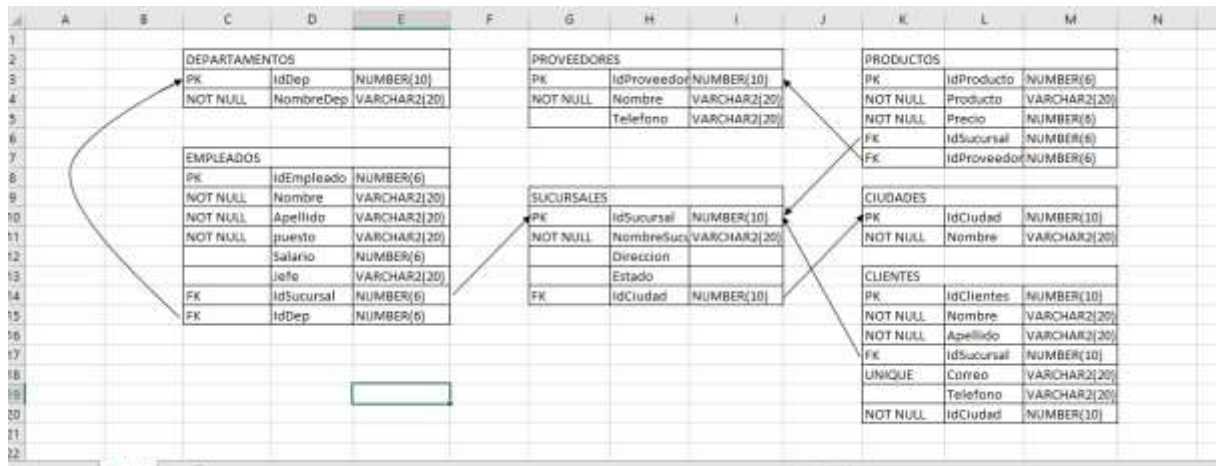
Se requiere la información de cada sucursal que poseen y por cada sucursal los datos de los empleados, clientes y productos van de la mano con su respectiva llave foránea. Tambien la tabla de productos lleva su llave foránea hacia la llave primaria de proveedores, asi como la llave foránea de sucursales va dirigida a la llave de ciudades y también la fk de empleados con la pk de departamentos

- ❖ Modelo de entidades (diagramas de tablas, ver Anexo 2)

- Mínimo de 7 tablas
- Nombre de la tabla
- Nombre de las columnas
- Tipos de datos de las columnas
- Restricciones de las columnas
- Comandos DDL para la creación de las tablas

(Fecha recomendada: 05/Ene/22)

Se crearon 7 tablas cada una con sus respectivas columnas, especificando su tipo de dato, todas con una primary key y algunas con foreign keys, también se añadieron columnas con datos con restricciones not null o unique



- ❖ Consultas INSERT para alimentar las tablas con datos (Mínimo 5 filas por tabla)
(Fecha recomendada: 10/Ene/22)

```

INSERT INTO departamentos VALUES (1, 'Gerente');
INSERT INTO departamentos VALUES (2, 'Cajero');
INSERT INTO departamentos VALUES (3, 'Almacen');
INSERT INTO departamentos VALUES (4, 'Limpieza');
INSERT INTO departamentos VALUES (5, 'Farmacia');

INSERT INTO proveedores VALUES (1, 'Alpura', '7284672040');
INSERT INTO proveedores VALUES (2, 'Pilgrims', '294629472');
INSERT INTO proveedores VALUES (3, 'Gamesa', '1092837465');
INSERT INTO proveedores VALUES (4, 'Bimbo', '926492530');
INSERT INTO proveedores VALUES (5, 'Cocacola', '529364472');

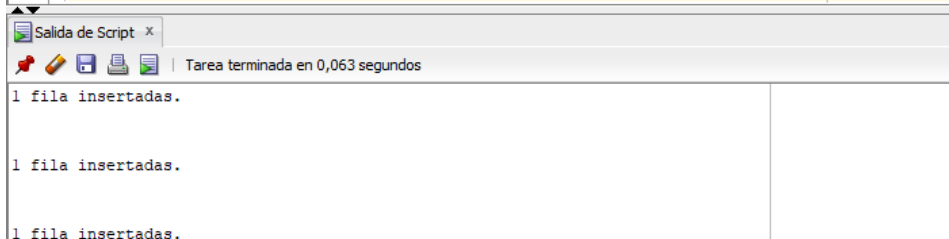
INSERT INTO ciudades VALUES (1, 'Queretaro');
INSERT INTO ciudades VALUES (2, 'CDMX');
INSERT INTO ciudades VALUES (3, 'Veracruz');
INSERT INTO ciudades VALUES (4, 'Monterrey');
INSERT INTO ciudades VALUES (5, 'Guadalajara');

INSERT INTO sucursales VALUES (1, 'Juriquilla', 'Fray JuniperoSerra', 'Queretaro', 1);
INSERT INTO sucursales VALUES (2, 'Universidad', 'Libre 450', 'CDMX', 2);
INSERT INTO sucursales VALUES (3, 'Norte', 'Rafael Cuervo', 'Veracruz', 3);
INSERT INTO sucursales VALUES (4, 'Las Torres', 'Av Eugenio Garza', 'Monterrey', 4);
INSERT INTO sucursales VALUES (5, 'Independencia', 'Av Periferico', 'Guadalajara', 5);

INSERT INTO clientes VALUES (1, 'Carlos', 'Correa', 1, 'ccorr@gmail.com', '7281638091', 1);
INSERT INTO clientes VALUES (2, 'Andrea', 'Ochoa', 2, 'aoch@gmail.com', '1956294620', 2);
INSERT INTO clientes VALUES (3, 'Jazmin', 'Morales', 3, 'jazzm@gmail.com', '9018273920', 3);
INSERT INTO clientes VALUES (4, 'Emiliano', 'Franco', 4, 'emfr@gmail.com', '2293828007', 4);
INSERT INTO clientes VALUES (5, 'Berenice', 'Tenorio', 5, 'beret@gmail.com', '3849382077', 5);

INSERT INTO empleados VALUES (1, 'Ivan', 'Morales', 'Gerente', 20500, 'NULL', 1, 1);
INSERT INTO empleados VALUES (2, 'Jorge', 'Espinosa', 'Cajero', 7500, 'Ivan', 2, 2);
INSERT INTO empleados VALUES (3, 'Omar', 'Diaz', 'Almacen', 10000, 'Ivan', 3, 3);
INSERT INTO empleados VALUES (4, 'Andres', 'Nieto', 'Limpieza', 8000, 'Ivan', 4, 4);
INSERT INTO empleados VALUES (5, 'Karen', 'Diaz', 'Farmacia', 12500, 'Ivan', 5, 5);

INSERT INTO productos VALUES (1, 'Galletas', 15, 1, 1);
INSERT INTO productos VALUES (2, 'Leche', 20, 2, 2);
INSERT INTO productos VALUES (3, 'Pan Blanco', 30, 3, 3);
INSERT INTO productos VALUES (4, 'Refresco', 25, 4, 4);
INSERT INTO productos VALUES (5, 'Pollo frito', 40, 5, 5);
  
```



- ❖ 3 consultas SELECT con cláusula ORDER BY (Fecha recomendada: 12/Ene/22)

ORDENAR LAS TABLAS EN ORDEN DE IDEMPLEADO

```
SELECT IdEmpleado, Nombre , Apellido
FROM empleados
ORDER BY IdEmpleado;
```

IDEMPLEADO	NOMBRE	APELLIDO
1	Ivan	Morales
2	Jorge	Espinosa
3	Omar	Diaz
4	Andres	Nieto
5	Karen	Diaz

ORDENAR POR SUCURSALES

```
SELECT IdSucursal, NombreSucursal, Estado
FROM sucursales
ORDER BY NombreSucursal;
```

IDSUCURSAL	NOMBRESUCURSAL	ESTADO
1	5 Independencia	Guadalajara
2	1 Juriquilla	Queretaro
3	4 Las Torres	Monterrey
4	3 Norte	Veracruz
5	2 Universidad	CDMX

ORDENAR POR IDSUCURSAL

```
SELECT Producto, Precio, idproducto
FROM PRODUCTOS
ORDER BY IdSucursal;
```

PRODUCTO	PRECIO	IDPRODUCTO
1 Galletas	15	1
2 Leche	20	2
3 Pan Blanco	30	3
4 Refresco	25	4
5 Pollo frito	40	5

❖ 3 consultas SELECT con condiciones lógicas (AND, OR, NOT) (Fecha recomendada: 12/Ene/22)

MOSTRAR EMPLEADO POR NOMBRE Y APELLIDO JORGE ESPINOSA

```
SELECT *
FROM EMPLEADOS
WHERE Apellido = 'Espinosa'
and Nombre = 'Jorge';
```

IDEMPLEADO	NOMBRE	APELLIDO	PUESTO	SALARIO	JEFE	IDSUCURSAL	IDDEP
1	2 Jorge	Espinosa	Cajero	7500	Ivan	2	2

MOSTRAR EMPLEADO POR NOMBRE O APELLIDO

```
SELECT *
FROM Clientes
WHERE Nombre = 'Jazmin'
OR Apellido = 'Tenorio';
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 2 en 0,003 segundos

IDCLIENTES	NOMBRE	APELLIDO	IDSUCURSAL	CORREO	TELEFONO	IDCIUDAD
1	3 Jazmin	Morales		3 jazm@gmail.com	9018273920	3
2	5 Berenice	Tenorio		5 beret@gmail.com	3849382077	5

MOstrar a todos los clientes pero excluya al cliente Carlos

```
SELECT *
FROM Clientes
WHERE NOT Nombre = 'Carlos';
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 4 en 0,006 segundos

IDCLIENTES	NOMBRE	APELLIDO	IDSUCURSAL	CORREO	TELEFONO	IDCIUDAD
1	2 Andrea	Ochoa		2 aoch@gmail.com	1956294620	2
2	3 Jazmin	Morales		3 jazm@gmail.com	9018273920	3
3	4 Emiliano	Franco		4 emfr@gmail.com	2293828007	4
4	5 Berenice	Tenorio		5 beret@gmail.com	3849382077	5

❖ 1 consulta SELECT con uso de:

- Operadores aritméticos

MOstrar SALARIO CON 20% DE COMISION CON PUESTO DE FARMACIA

```
SELECT nombre, (salario * .2) AS Salario_total
FROM empleados
WHERE puesto='Farmacia';
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,004 segundos

	NOMBRE	SALARIO_TOTAL
1	Karen	2500

- Operadores de concatenación

CONCATENA EL NOMBRE Y APELLIDO DE EMPLEADOS

```
SELECT CONCAT( Nombre, Apellido ) as NombreCompleto
FROM EMPLEADOS;
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 5 en 0,002 segundos

	NOMBRECOMPLETO
1	IvanMorales
2	JorgeEspinosa
3	OmarDiaz
4	AndresNieto
5	KarenDiaz

- Condiciones de búsqueda

MOstrar APELLIDOS QUE NO TENGAN LA LETRA R

<pre>SELECT IdEmpleado, Nombre, Apellido FROM EMPLEADOS WHERE upper(Apellido) NOT LIKE '%R%';</pre>			
<div>Salida de Script x</div> <div>Resultado de la Consulta x</div> <div>SQL Todas las Filas Recuperadas: 4 en 0,002</div>			
IDEMPLEADO	NOMBRE	APELLIDO	
1	2 Jorge	Espinosa	
2	3 Omar	Diaz	
3	4 Andres	Nieto	
4	5 Karen	Diaz	

- Uso de cláusula BETWEEN

MOSTRAR SALARIOS ENTRE 9000 Y 20000

<pre>SELECT Salario FROM EMPLEADOS WHERE Salario BETWEEN 9000 AND 20000;</pre>	
<div>Salida de Script x</div> <div>Resultado de la Consulta x</div> <div>SQL Todas las Filas Recuperadas: 2 en 0</div>	
SALARIO	
1	10000
2	12500

- Uso de condición NOT

<pre>SELECT Puesto, Salario FROM EMPLEADOS WHERE puesto NOT IN ('idempleado');</pre>	
<div>Salida de Script x</div> <div>Resultado de la Consulta x</div> <div>SQL Todas las Filas Recuperadas: 5 en</div>	
PUESTO	SALARIO
1 Gerente	20500
2 Cajero	7500
3 Almacen	10000
4 Limpieza	8000
5 Farmacia	12500

- Alias para las columnas

<pre>SELECT IdEmpleado as numero FROM EMPLEADOS;</pre>	
<div>Salida de Script x</div> <div>Resultado de la Consulta x</div> <div>SQL Todas las Filas Recuperadas:</div>	
NUMERO	
1	1
2	2
3	3
4	4
5	5

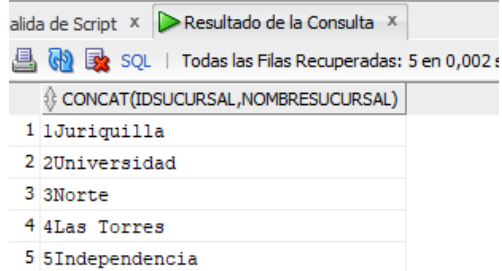
(Fecha recomendada: 12/Ene/22)

❖ 1 consulta SELECT con uso de:

- Función CONCAT

CONCATENA ID Y NOMBRE

```
SELECT CONCAT(IdSucursal, NombreSucursal)
FROM Sucursales;
```



Salida de Script x Resultado de la Consulta x

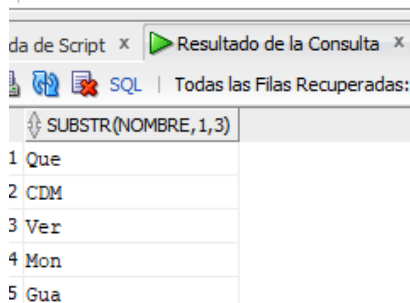
SQL | Todas las Filas Recuperadas: 5 en 0,002 s

	CONCAT(IDSUCURSAL,NOMBRESUCURSAL)
1	1Juriquilla
2	2Universidad
3	3Norte
4	4Las Torres
5	5Independencia

- Función SUBSTR

ABREVEA LOS NOMBRES DE CIUDADES CON LAS 3 PRIMERAS LETRAS

```
SELECT SUBSTR (Nombre, 1 , 3)
FROM ciudades;
```



Salida de Script x Resultado de la Consulta x

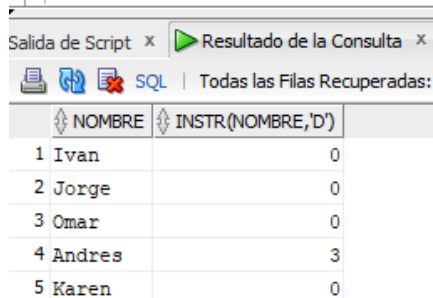
SQL | Todas las Filas Recuperadas:

	SUBSTR(NOMBRE,1,3)
1	Que
2	CDM
3	Ver
4	Mon
5	Gua

- Función INSTR

MUESTRA LA UBICACIÓN DE LA LETRA D EN TODOS LOS EMPLEADOS

```
SELECT Nombre, INSTR (Nombre, 'd')
FROM EMPLEADOS;
```



Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas:

	NOMBRE	INSTR(NOMBRE,'D')
1	Ivan	0
2	Jorge	0
3	Omar	0
4	Andres	3
5	Karen	0

(Fecha recomendada: 17/Ene/22)

❖ 1 consulta SELECT con uso de:

- Función LENGTH

MOSTRAR LONGITUD DE CARACTERES DE CADA PRODUCTO

```
SELECT Producto, LENGTH (Producto)
FROM Productos;
```

PRODUCTO	LENGTH(PRODUCTO)
1 Galletas	8
2 Leche	5
3 Pan Blanco	10
4 Refresco	8
5 Pollo frito	11

○ Función TRIM

USE UN TRIM PARA ELIMINAR ESPACIO EN BLANCO DEL CLIENTE CARLOS

```
SELECT TRIM (' Carlos')
FROM clientes;
```

TRIM(CARLOS)
1 Carlos

○ Función ROUND

HAY UNA COMISION DE 1547,349 REDONDEE ESTA CANTIDAD A 1 DECIMA

```
SELECT round (1547.349, 1)
FROM dual;
```

ROUND(1547.349,1)
1 1547,3

○ Función TRUNC

TRUNCA EL PRECIO DE PRODUCTOS DE 15 PESOS PARA AGREGAR EL IVA DESPUES

```
SELECT TRUNC (15.79,0) "IVA"
FROM PRODUCTOS;
```

IVA
1 15

(Fecha recomendada: 17/Ene/22)

✦ 1 consulta SELECT con uso de:

○ Función MONTHS_BETWEEN

MUESTRE LOS MESES ENTRE HOY Y EL INICIO DE AÑO PARA VER CUANTO TIEMPO LLEVAN OPERANDO LAS SUCURSALES

```
SELECT MONTHS_BETWEEN(Sysdate,'01-01-2022')
FROM sucursales;
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 5 en 0 segundos

	MONTHS_BETWEEN(SYSDATE,'01-01-2022')
1	1,21850134408602150537634408602150537634
2	1,21850134408602150537634408602150537634
3	1,21850134408602150537634408602150537634
4	1,21850134408602150537634408602150537634
5	1,21850134408602150537634408602150537634

○ Función NEXT_DAY

MUESTRE EL SIGUIENTE LUNES LABORAL

```
SELECT NEXT_DAY (SYSDATE, 'LUNES') FROM DUAL;
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,003 segundos

	NEXT_DAY(SYSDATE,'LUNES')
1	14/02/22

○ Función LAST_DAY

MUESTRA EL ULTIMO DIA DE ESTE MES PARA HACER CUENTAS

```
SELECT LAST_DAY (SYSDATE) FROM DUAL;
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0 segundos

	LAST_DAY(SYSDATE)
1	28/02/22

(Fecha recomendada: 17/Ene/22)

❖ 1 consulta SELECT con uso de:

○ Función AVG

MUESTRA EL PROMEDIO DEL SALARIO DE TODOS LOS EMPLEADOS

```
SELECT AVG(Salario)
FROM Empleados;
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0 segundos

	AVG(SALARIO)
1	11700

○ Función MAX

MUESTRA EL PRECIO DEL PRODUCTO CON MAYOR COSTO

```
SELECT MAX(Precio)
FROM Productos;
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0 segundos

	MAX(PRECIO)
1	40

○ Función MIN

MUESTRA EL PRECIO DEL PRODUCTO CON MENOR COSTO


```
SELECT MIN(Precio)
FROM Productos;
```

MIN(PRECIO)
15

○ Cláusula GROUP BY

MUESTRE EL ID DE DEPARTAMENTO, SALARIO PROMEDIO Y AGRUPA POR ID DE DEPT

```
SELECT IdDep , AVG(Salario)
FROM EMPLEADOS
GROUP BY IdDep;
```

IDDEP	AVG(SALARIO)
1	20500
2	7500
3	8000
4	12500
5	10000

(Fecha recomendada: 17/Ene/22)

- ❖ 1 consulta SELECT con uso de NATURAL JOIN entre 2 tablas (Fecha recomendada: 19/Ene/22)
CONECTE SUCURSALES Y CIUDADES PARA SABER IDCIUDAD Y IDSUCURSAL

```
SELECT IdCiudad, IdSucursal
FROM Sucursales
NATURAL JOIN ciudades;
```

IDCIUDAD	IDSUCURSAL
1	1
2	2
3	3
4	4
5	5

- ❖ 1 consulta SELECT con uso de JOIN entre 3 tablas (Fecha recomendada: 19/Ene/22)
SELECCIONA IDSUCURSAL, APELLIDO, NOMBRE DE CIUDAD DONDE EL PUESTO SEA GERENTE, UTILIZA JOIN ENTRE LAS 3 TABLAS

```
SELECT s.IdSucursal, Emp.Apellido, ci.nombre as Ciudad
FROM sucursales s
JOIN EMPLEADOS Emp
ON s.IdSucursal=Emp.IdSucursal
JOIN ciudades ci
ON s.IdCiudad=ci.IdCiudad
WHERE Emp.Puesto='Gerente';
```

IDSUCURSAL	APELLIDO	CIUDAD
1	1 Morales	Queretaro

- ❖ 2 consultas SELECT con uso de subconsultas (Fecha recomendada: 19/Ene/22)
MUESTRA IDEMPLEADO Y APELLIDO QUE TRABAJEN EN EL DEPT DE CUALQUIER

EMPLEADO QUE SU APELLIDO TIENE UNA LETRA A

```
SELECT IdEmpleado, Apellido FROM Empleados
WHERE IdEmpleado IN (SELECT IdDep
FROM Empleados
WHERE Apellido like '%a%');
```

Resultado de la Consulta	
Todas las Filas Recuperadas: 4 en 0,035 segundos	
IDEMPLEADO	APELLIDO
1	1 Morales
2	2 Espinosa
3	3 Diaz
4	5 Diaz

MUESTRE LOS DATOS DE LOS EMPLEADOS DEL DEPT DE FARMACIA

```
SELECT IdDep, Apellido, Puesto FROM Empleados
WHERE Puesto IN (SELECT Puesto
FROM Empleados
WHERE Puesto = 'Farmacia');
```

Resultado de la Consulta		
Todas las Filas Recuperadas: 1 en 0,016 segundos		
IDEP	APELLIDO	PUESTO
1	5 Diaz	Farmacia

- ❖ 1 consulta SELECT con operador UNION (Fecha recomendada: 24/Ene/22)
UNE EMPLEADOS Y DEPARTAMENTOS POR EL IDDEP

```
SELECT IdDep, Nombre
FROM Empleados
UNION
SELECT IdDep, NombreDep
FROM Departamentos;
```

Resultado de la Consulta	
Todas las Filas Recuperadas: 10 en 0,016 segundos	
IDDEP	NOMBRE
1	1 Gerente
2	1 Ivan
3	2 Cajero
4	2 Jorge
5	3 Almacen
6	3 Omar
7	4 Andres
8	4 Limpieza
9	5 Farmacia
10	5 Karen

- ❖ Consultas SQL para la creación de 3 VIEWS (Fecha recomendada: 24/Ene/22)

```
CREATE VIEW DatosGenerales AS
SELECT Nombre, Apellido
FROM empleados;
CREATE VIEW DatosGeneralesCiudad AS
SELECT Nombre, IdCiudad
FROM Ciudades;
CREATE VIEW DatosGeneralesSucursal
AS SELECT NombreSucursal, Estado
FROM Sucursales;
```

Salida de Script x Resultado de la Consulta x
Tarea terminada en 0,029 segundos

View DATOSGENERALES creado.

View DATOSGENERALESCIUDAD creado.

View DATOSGENERALESSUCURSAL creado.

- ❖ Los valores de llaves primarias (PK) de las tablas deben obtenerse de una SEQUENCE
 - Crear 1 SEQUENCE por cada tabla para obtener su PK

(Fecha recomendada: 26/Ene/22)

```
CREATE SEQUENCE "SECUENCIA_CLIENTES" MINVALUE 1 MAXVALUE 1000
INCREMENT BY 10 START WITH 200 NOCACHE NOORDER NOCYCLE;
CREATE SEQUENCE "SECUENCIA_PRODUCTOS" MINVALUE 1 MAXVALUE 1000
INCREMENT BY 10 START WITH 200 NOCACHE NOORDER NOCYCLE;
CREATE SEQUENCE "SECUENCIA_EMPLEADOS" MINVALUE 1 MAXVALUE 1000
INCREMENT BY 10 START WITH 200 NOCACHE NOORDER NOCYCLE;
CREATE SEQUENCE "SECUENCIA_CIUDADES" MINVALUE 1 MAXVALUE 1000
INCREMENT BY 10 START WITH 200 NOCACHE NOORDER NOCYCLE;
CREATE SEQUENCE "SECUENCIA_DEPT" MINVALUE 1 MAXVALUE 1000
INCREMENT BY 10 START WITH 200 NOCACHE NOORDER NOCYCLE;
CREATE SEQUENCE "SECUENCIA_PROVEEDORES" MINVALUE 1 MAXVALUE 1000
INCREMENT BY 10 START WITH 200 NOCACHE NOORDER NOCYCLE;
CREATE SEQUENCE "SECUENCIA_SUCURSALES" MINVALUE 1 MAXVALUE 1000
INCREMENT BY 10 START WITH 200 NOCACHE NOORDER NOCYCLE;
```

Salida de Script x
Tarea terminada en 0,062 segundos

Sequence "SECUENCIA_CLIENTES" creado.

Sequence "SECUENCIA_PRODUCTOS" creado.

Sequence "SECUENCIA_EMPLEADOS" creado.

Sequence "SECUENCIA_CIUDADES" creado.

Sequence "SECUENCIA_DEPT" creado.

Consultas SQL para la creación de 2 SYNONYMS (Fecha recomendada: 26/Ene/22)

```
CREATE SYNONYM NOMSUC
FOR SUCURSALES_NOMBRESUCURSAL;

CREATE SYNONYM IDSUC
FOR SUCURSALES_IDSUCURSAL;
```

Salida de Script x
Tarea terminada en 0,043 segundos

Synonym NOMSUC creado.

Synonym IDSUC creado.

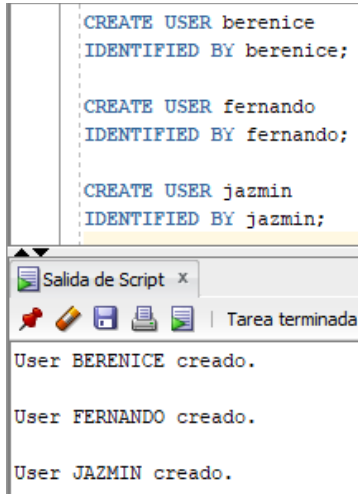
Consultas SQL para la creación de 3 usuarios diferentes (Fecha recomendada:

26/Ene/22)

```
CREATE USER berenice
IDENTIFIED BY berenice;

CREATE USER fernando
IDENTIFIED BY fernando;

CREATE USER jazmin
IDENTIFIED BY jazmin;
```



Salida de Script x

Tarea terminada

User BERENICE creado.

User FERNANDO creado.

User JAZMIN creado.

Consultas SQL para asignar a cada uno de los 3 usuarios anteriores los privilegios de: ○ INICIAR SESIÓN

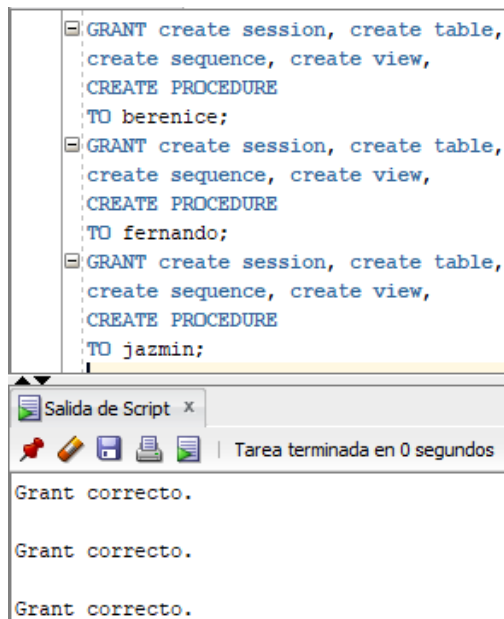
- CREAR TABLAS
- CREAR VISTAS
- CREAR SECUENCIAS
- OBTENER ESPACIO DE ALMACENAMIENTO (QUOTA)

(Fecha recomendada: 31/Ene/22)

```
GRANT create session, create table,
create sequence, create view,
CREATE PROCEDURE
TO berenice;

GRANT create session, create table,
create sequence, create view,
CREATE PROCEDURE
TO fernando;

GRANT create session, create table,
create sequence, create view,
CREATE PROCEDURE
TO jazmin;
```



Salida de Script x

Tarea terminada en 0 segundos

Grant correcto.

Grant correcto.

Grant correcto.

Consulta SQL para crear un BLOQUE ANÓNIMO que imprima todos los datos de una tabla mediante un CURSOR y un BUCLE (Fecha recomendada: 31/Ene/22)

```

DECLARE
CURSOR cEmpleado IS
SELECT IdEmpleado, Apellido, Salario
FROM EMPLEADOS;
rEmpleado cEmpleado%ROWTYPE;
BEGIN
OPEN cEmpleado;
FETCH cEmpleado INTO rEmpleado;
WHILE cEmpleado%FOUND LOOP
dbms_output.put_line(rEmpleado.IdEmpleado || ' '
|| rEmpleado.Apellido
|| ' ' || rEmpleado.Salario);
FETCH cEmpleado INTO rEmpleado;
END LOOP;
CLOSE cEmpleado;
END;

```

Salida de Script x

Tarea terminada en 0 segundos

Procedimiento PL/SQL terminado correctamente.

```

1 Morales 20500
2 Espinosa 7500
3 Diaz 10000
4 Nieto 8000
5 Diaz 12500

```

Consultas SQL para la creación de un PROCEDIMIENTO ALMACENADO que tome 1 parámetro de entrada (Fecha recomendada: 31/Ene/22)

PROCEDIMIENTO ALMACENADO QUE AUMENTE EL SALARIO 10% EN UNA FILA

The screenshot shows the Oracle SQL Developer interface. On the left, the 'sams' schema is expanded, showing various database objects. The 'Procedimientos' (Procedures) folder is selected, and a new procedure named 'AUMENTO' is being created. The main window displays the SQL script for the procedure, which takes an input parameter 'v_id' and updates the salary of the employee with that ID by 10%. The 'Salida de Script' (Script Output) window at the bottom shows that the procedure was compiled successfully and executed without errors.

```

CREATE OR REPLACE PROCEDURE Aumento
(v_id IN Empleados.IdEmpleado%TYPE) IS
BEGIN
UPDATE Empleados SET Salario = Salario * 1.10
WHERE IdEmpleado = v_id;
END Aumento;

EXECUTE Aumento (5);

```

Salida de Script x

Tarea terminada en 0,063 segundos

Procedure AUMENTO compilado

Procedimiento PL/SQL terminado correctamente.

Consultas SQL para la creación de un PROCEDIMIENTO ALMACENADO que tome 1 parámetro de entrada y 1 parámetro de salida (Fecha recomendada: 31/Ene/22)

POR MEDIO DEL ID DE UN EMPLEADO MOSTRAR NOMBRE Y SALARIO

```

CREATE OR REPLACE PROCEDURE
BUSCAR
(v_id IN NUMBER,
v_nombre out
EMPLEADOS.Nombre%TYPE ,
v_sal OUT number )
is begin
select Nombre, Salario
into v_nombre, v_sal
from EMPLEADOS
where IdEmpleado = v_id;
END BUSCAR;
VARIABLE g_name VARCHAR2(20);
VARIABLE g_salary NUMBER;
EXECUTE BUSCAR(5, :g_name, :g_salary);
PRINT g_name;
PRINT g_salary;

```

Salida de Script x

Tarea terminada en 0,047 segundos

Procedure BUSCAR compilado

Procedimiento PL/SQL terminado correctamente.

G_NAME

Karen

G_SALARY

13750

Consultas SQL para la creación de un PROCEDIMIENTO ALMACENADO que tome 1 parámetro de entrada y salida. (Fecha recomendada: 02/Feb/22)

PROCEDIMIENTO PARA DARLE ESPACIAMIENTO A UN NUMERO TELEFONICO

```

CREATE OR REPLACE PROCEDURE
EJEMPLO3
(v_phone_no IN OUT VARCHAR2) IS
BEGIN
v_phone_no := '(' ||
SUBSTR(v_phone_no,1,3)
|| ')' || SUBSTR(v_phone_no,4,3) ||
'-' || SUBSTR(v_phone_no,7);
END EJEMPLO3;
VARIABLE g_phone_no varchar2(15);
BEGIN
:g_phone_no := '4425974121';
END;
EXECUTE EJEMPLO3(:g_phone_no);
PRINT g_phone_no;

```

Salida de Script x

Tarea terminada en 0,032 segundos

Procedure EJEMPLO3 compilado

Procedimiento PL/SQL terminado correctamente.

Procedimiento PL/SQL terminado correctamente.

G_PHONE_NO

(442) 597-4121

Consultas SQL para la creación de 3 FUNCIONES (Fecha recomendada: 02/Feb/22)

TOMAR EL SALARIO DE UN EMPLEADO QUE SU ID CORRESPONDA A V_SALARIO

```
CREATE OR REPLACE FUNCTION FUNCION1
(v_id IN EMPLEADOS.IdEmpleado%TYPE)
RETURN NUMBER IS v_salario EMPLEADOS.Salario%TYPE :=0;
BEGIN
SELECT Salario
INTO v_salario
FROM EMPLEADOS
WHERE IdEmpleado = v_id;
RETURN (v_salario);
END FUNCION1;

VARIABLE FUNCION1 number
BEGIN :FUNCION1 :=FUNCION1(5); END;

PRINT FUNCION1;
```

Salida de Script x

Tarea terminada en 0,025 segundos

Function FUNCION1 compilado

SP2-0552: La variable de enlace "FUNCION1" no está declarada.
Procedimiento PL/SQL terminado correctamente.
FUNCION1

13750

CONSULTA EL DEPT DE UN EMPLEADO POR MEDIO DE SU ID

```
CREATE OR REPLACE FUNCTION FUNCION2
(v_id IN Empleados.IdEmpleado%TYPE)
RETURN VARCHAR2 IS
v_DEP
DEPARTAMENTOS.NombreDep %TYPE;
BEGIN
SELECT NombreDep
INTO v_DEP
FROM EMPLEADOS
JOIN DEPARTAMENTOS
USING (IdDep)
WHERE IdEmpleado = v_id;
RETURN (v_DEP);
END FUNCION2;

VARIABLE FUNCION2 NUMBER;
BEGIN :FUNCION2 :=FUNCION2(5); END;

PRINT FUNCION2;
```

Salida de Script x

Tarea terminada en 0,032 segundos

Function FUNCION2 compilado

Procedimiento PL/SQL terminado correctamente.
FUNCION2

Farmacia

```
CREATE OR REPLACE FUNCTION
FUNCION3
(v_valor IN NUMBER)
RETURN NUMBER
IS BEGIN
RETURN (v_valor *0.08);
END FUNCION3;
VARIABLE FUNCION3 NUMBER;
BEGIN :FUNCION3 :=FUNCION3(5);END;
PRINT FUNCION3;
```

Salida de Script x

Tarea terminada en 0,016 segundos

Function FUNCION3 compilado

Procedimiento PL/SQL terminado correctamente.
FUNCION3

,4

Consultas SQL para la creación de 3 TRIGGERS (Fecha recomendada: 09/Feb/22)
TRIGGER QUE SOLO ADMITA MODIFICAR EMPLEADOS EN HORARIOS LABORALES

```
CREATE OR REPLACE TRIGGER HorarioCambios
BEFORE INSERT ON Empleados
BEGIN
IF (TO_CHAR(sysdate, 'DY') IN ('SAT', 'SUN'))
OR (TO_CHAR(sysdate, 'HH24') NOT BETWEEN '08' AND '10')
THEN RAISE_APPLICATION_ERROR(-20500, 'Modificaciones solo en horario laboral');
END IF;
END;
```

Salida de Script x

Tarea terminada en 0,298 segundos

Trigger HORARIOCAMBIOS compilado

TRIGGER PARA VALIDAR INSERCIONES O ACTUALIZACIONES EN EMPLEADOS,
QUE DATOS SEAN MAYOR A 0, SI NO QUE SE LE ASIGNE 0 ANTES DE ALGUN
UPDATE

```
CREATE OR REPLACE TRIGGER Valida
BEFORE INSERT OR UPDATE ON Empleados
FOR EACH ROW
BEGIN
IF (:new.Salario < 0) OR :new.Salario IS NULL THEN
:new.Salario :=10;
END IF;
END;
```

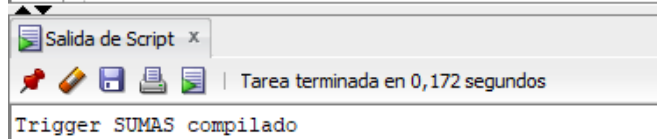
Salida de Script x

Tarea terminada en 0,062 segundos

Trigger VALIDA compilado

TRIGGER PARA MOSTRAR SUMA DE SALARIOS CADA QUE SE EJECUTE UNA
OPERACIÓN DML EN EMPLEADOS


```
CREATE OR REPLACE TRIGGER SUMAS
AFTER INSERT OR UPDATE OR DELETE ON EMPLEADOS
DECLARE v_sal NUMBER;
BEGIN
SELECT SUM(Salario)
INTO v_sal
FROM Empleados;
INSERT INTO resultados VALUES (user, v_sal);
END;
```



Conclusiones personales acerca del proyecto (Fecha recomendada: 09/Feb/22)

El proyecto te lleva por todos los temas que se vieron en el diplomado desde el principio por lo que al estar bien organizado y gracias a las practicas que se hicieron a lo largo de estos meses uno puede desarrollar su propio proyecto por su cuenta, desde la creación de tablas hasta los triggers.

El objetivo del proyecto era crear una base de datos para la empresa de tiendas SAMS CLUB, este objetivo se pudo cumplir a partir del desarrollo del conocimiento adquirido a lo largo del diplomado el cual el maestro explico con claridad y buena organización.

Este diplomado ha hecho un cambio en mi forma de comprender las bases de datos y también me abre a saber que hay demasiadas aplicaciones en el mundo laboral así como cada tema del tema se puede profundizar y especializar.

Finalmente me queda agradecer por el aprendizaje obtenido y el como pude reafirmar los conocimientos que ya tenia así como el entender como hacer un trabajo de este tamaño y el proceso que se necesita para realizarlo.

NOTA: Todas las consultas SELECT y el código PL/SQL deben ir acompañados de un texto descriptivo que detalle su funcionalidad, así como una captura de pantalla del resultado.

Fecha de Entrega de Proyecto Final

09 de febrero de 2022.

Formato de Entrega de Proyecto Final

Subir este documento contestado en formato PDF al apartado correspondiente del Classroom.

Requisitos de Acreditación del Diplomado

100% de prácticas entregadas

Proyecto Final entregado en tiempo y forma