EVALUACIÓN PL/SQL - DESARROLLADOR JUNIOR/SEMI-SENIOR

REGLAS GENERALES

- Base de datos: Oracle 11gR2 o superior
- Organización: Todo el código debe estar en archivos .sql numerados
- Entrega: Repositorio GitHub público con archivos ordenados
- Tiempo: 2 días (más tiempo para aprender)
- Ayuda: Se permite consultar documentación de Oracle

CONTEXTO

Una agremiación de comercio necesita un sistema básico para gestionar información de comerciantes y sus establecimientos.

RETO 01 - CREAR TABLAS BÁSICAS ⭐

Crear tablas para almacenar información de:

Tabla COMERCIANTES:

```
sql
- ID (número, clave primaria)
- NOMBRE (texto, obligatorio)
- DEPARTAMENTO (texto, obligatorio)
- MUNICIPIO (texto, obligatorio)
- TELEFONO (texto, opcional)
- CORREO (texto, opcional)
- FECHA_REGISTRO (fecha, obligatorio)
- ESTADO (texto: 'ACTIVO', 'INACTIVO', 'SUSPENDIDO')
- FECHA_ACTUALIZACION (fecha)
- USUARIO_ACTUALIZACION (texto)
```

Tabla ESTABLECIMIENTOS:

```sql
- ID (número, clave primaria)
- COMERCIANTE_ID (número, referencia a comerciantes)
- NOMBRE (texto, obligatorio)
- INGRESOS (número con 2 decimales)
- NUM_EMPLEADOS (número entero)
- FECHA_ACTUALIZACION (fecha)
- USUARIO_ACTUALIZACION (texto)
```

Entregable: `01_crear_tablas.sql`

RETO 02 - SECUENCIAS Y TRIGGERS BÁSICOS ⭐⭐

Crear secuencias para IDs:

```sql
CREATE SEQUENCE SEQ_COMERCIANTES START WITH 1;
CREATE SEQUENCE SEQ_ESTABLECIMIENTOS START WITH 1;
```

Triggers simples para IDs automáticos:

- Trigger que asigne automáticamente el ID al insertar comerciantes
- Trigger que asigne automáticamente el ID al insertar establecimientos

Triggers de auditoría básicos:

- Actualizar FECHA_ACTUALIZACION y USUARIO_ACTUALIZACION en INSERT/UPDATE

Entregable: `02_secuencias_triggers.sql`

RETO 03 - DATOS DE PRUEBA ⭐

Insertar datos semilla:

- 5 comerciantes con diferentes estados
- 15 establecimientos distribuidos entre los comerciantes

- Usar datos realistas (nombres de empresas, departamentos de Colombia)

Entregable: `03_datos_prueba.sql`

RETO 04 - CONSULTAS BÁSICAS CON JOINS ⭐⭐

Crear vistas simples para consultas frecuentes:

Vista: Comerciantes con totales

```sql
CREATE OR REPLACE VIEW V_COMERCIANTES_RESUMEN AS
SELECT
    c.ID,
    c.NOMBRE,
    c.DEPARTAMENTO,
    c.MUNICIPIO,
    c.TELEFONO,
    c.CORREO,
    c.FECHA_REGISTRO,
    c.ESTADO,
    COUNT(e.ID) as CANTIDAD_ESTABLECIMIENTOS,
    NVL(SUM(e.INGRESOS), 0) as TOTAL_INGRESOS,
    NVL(SUM(e.NUM_EMPLEADOS), 0) as TOTAL_EMPLEADOS
FROM COMERCIANTES c
LEFT JOIN ESTABLECIMIENTOS e ON c.ID = e.COMERCIANTE_ID
GROUP BY c.ID, c.NOMBRE, c.DEPARTAMENTO, c.MUNICIPIO,
         c.TELEFONO, c.CORREO, c.FECHA_REGISTRO, c.ESTADO;
```

Entregable: `04_vistas_consultas.sql`

RETO 05 - PROCEDIMIENTOS ALMACENADOS SIMPLES ⭐⭐

Crear procedimientos básicos:

1. Crear comerciante:

```sql
CREATE OR REPLACE PROCEDURE SP_CREAR_COMERCIANTE(
    P_NOMBRE IN VARCHAR2,
    P_DEPARTAMENTO IN VARCHAR2,
    P_MUNICIPIO IN VARCHAR2,
    P_TELEFONO IN VARCHAR2 DEFAULT NULL,
    P_CORREO IN VARCHAR2 DEFAULT NULL,
    P_ESTADO IN VARCHAR2 DEFAULT 'ACTIVO',
    P_RESULTADO OUT NUMBER,
    P_MENSAJE OUT VARCHAR2
);
```

2. Actualizar comerciante:

```sql
CREATE OR REPLACE PROCEDURE SP_ACTUALIZAR_COMERCIANTE(
    P_ID IN NUMBER,
    P_NOMBRE IN VARCHAR2,
    P_DEPARTAMENTO IN VARCHAR2,
    P_MUNICIPIO IN VARCHAR2,
    P_TELEFONO IN VARCHAR2 DEFAULT NULL,
    P_CORREO IN VARCHAR2 DEFAULT NULL,
    P_RESULTADO OUT NUMBER,
    P_MENSAJE OUT VARCHAR2
);
```

3. Eliminar comerciante (lógico):

```sql
CREATE OR REPLACE PROCEDURE SP_ELIMINAR_COMERCIANTE(
    P_ID IN NUMBER,
    P_RESULTADO OUT NUMBER,
    P_MENSAJE OUT VARCHAR2
);
```

Validaciones básicas:

- Campos obligatorios no nulos
- Formato de correo válido (contiene @)

- Estados válidos

Entregable: `05_procedimientos_comerciantes.sql`

RETO 06 - FUNCIONES SIMPLES ⭐⭐

Crear funciones para consultas:

1. Obtener comerciante por ID:

```
sql
CREATE OR REPLACE FUNCTION FN_OBTENER_COMERCIANTE(P_ID IN
NUMBER)
RETURN SYS_REFCURSOR;
```

2. Listar comerciantes con filtros:

```
sql
CREATE OR REPLACE FUNCTION FN_LISTAR_COMERCIANTES(
    P_NOMBRE IN VARCHAR2 DEFAULT NULL,
    P_MUNICIPIO IN VARCHAR2 DEFAULT NULL,
    P_ESTADO IN VARCHAR2 DEFAULT NULL
) RETURN SYS_REFCURSOR;
```

3. Contar total de comerciantes:

```
sql
CREATE OR REPLACE FUNCTION FN_CONTAR_COMERCIANTES
RETURN NUMBER;
```

Entregable: `06_funciones_consulta.sql`

RETO 07 - CRUD ESTABLECIMIENTOS ⭐⭐

Procedimientos para establecimientos:

1. Crear establecimiento:

```
sql
```

```
CREATE OR REPLACE PROCEDURE SP_CREAR_ESTABLECIMIENTO(
    P_COMERCIANTE_ID IN NUMBER,
    P_NOMBRE IN VARCHAR2,
    P_INGRESOS IN NUMBER,
    P_NUM_EMPLEADOS IN NUMBER,
    P_RESULTADO OUT NUMBER,
    P_MENSAJE OUT VARCHAR2
);
```

2. Listar establecimientos por comerciante:

```
sql
CREATE OR REPLACE FUNCTION
FN_ESTABLECIMIENTOS_COMERCIANTE(P_COMERCIANTE_ID IN NUMBER)
RETURN SYS_REFCURSOR;
```

Entregable: `07_crud_establecimientos.sql`

RETO 08 - REPORTES BÁSICOS ⭐

Crear función de reporte:

Función que genere cursor con comerciantes activos:

```
sql
CREATE OR REPLACE FUNCTION FN_REPORTE_COMERCIANTES_ACTIVOS
RETURN SYS_REFCURSOR;
```

Estructura del reporte:

- Nombre, Departamento, Municipio
- Teléfono, Correo, Fecha Registro
- Cantidad Establecimientos, Total Ingresos, Total Empleados
- Solo comerciantes ACTIVOS
- Ordenado por cantidad de establecimientos (mayor a menor)

Entregable: `08_reportes.sql`

RETO 09 - TESTING BÁSICO ⭐

Script de pruebas:

sql

```sql
-- Probar creación de comerciantes
-- Probar creación de establecimientos
-- Probar consultas
-- Probar reportes
-- Verificar que los triggers funcionen
```

Entregable: `09_testing.sql`

ENTREGABLES FINALES

Archivos requeridos (en orden):

1. `01_crear_tablas.sql`
2. `02_secuencias_triggers.sql`
3. `03_datos_prueba.sql`
4. `04_vistas_consultas.sql`
5. `05_procedimientos_comerciantes.sql`
6. `06_funciones_consulta.sql`
7. `07_crud_establecimientos.sql`
8. `08_reportes.sql`
9. `09_testing.sql`
10. `README.md` (explicando cómo ejecutar los scripts)

README.md debe incluir:

text

```
# Sistema Comerciantes - Base de Datos

## Orden de ejecución:
1. Ejecutar scripts en orden numérico
2. Verificar que no hay errores
3. Ejecutar script de testing

## Usuarios de prueba creados:
```

```
- 5 comerciantes de ejemplo
- 15 establecimientos distribuidos

## Funcionalidades:
- CRUD completo de comerciantes
- CRUD básico de establecimientos
- Reportes de comerciantes activos
- Auditoría automática
```

CRITERIOS DE EVALUACIÓN SIMPLIFICADOS

Aspecto	**Peso**	**Nivel Esperado**
Creación de tablas	20%	Correcto con PKs y FKs
Triggers básicos	15%	Funcionando correctamente
Procedimientos CRUD	25%	Con validaciones básicas
Consultas y reportes	20%	Joins correctos
Datos de prueba	10%	Realistas y completos
Organización código	10%	Scripts ordenados