

**PRODUCTO INTEGRADOR DE APRENDIZAJE**

Nombre de la Unidad de Aprendizaje: PROGRAMACION DE BASE DE DATOS

Nombre del proyecto: Código de PL/SQL de estructuras de datos con objetos complementarios de un caso de negocios planteado por el profesor.

Programa educativo: Licenciado en Tecnologías de Información.

Semestre: 3ro Grupo: 32

Nombre del maestro: M.C. LAURA ALICIA HERNANDEZ MORENO

Nombre de los integrantes del equipo:

(En orden alfabético apellidos-nombres- # matrícula)

Mata Coronado Orlando Fabian – 2012344

Melendez Aguilera Daniela – 1955973

|  |  |
| --- | --- |
| **Contenido mínimo a evaluar** | **Cumplimiento** |
| Índice |  |
| Introducción. - incluye valores UANL aplicados |  |
| Análisis y emisión de juicio |  |
| Conclusiones individuales |  |
| Conclusión del equipo |  |
| Actividad en inglés |  |
| Identificación de sub resultados de aprendizaje ANECA. |  |
| Calificación PIA: |  |
| Firma del maestro | |

|  |
| --- |
| ☐  Las personas que aparecen en esta portada aceptamos que hemos desarrollado este trabajo como creación propia y estamos conscientes de las reglas que se aplicarán en esta materia en caso de detectarse lo contrario o plagio. |

San Nicolás de los Garza, ciudad universitaria a (17-11-2024)

Índice

[Introducción (ANECA 2.1) 4](#_Toc182752521)

[Código DDL 5](#_Toc182752522)

[Código: 5](#_Toc182752523)

[Problema 1 9](#_Toc182752524)

[Código: 9](#_Toc182752525)

[Capturas: 14](#_Toc182752526)

[Mata Coronado Orlando Fabián 14](#_Toc182752527)

[Melendez Aguilera Daniela: 19](#_Toc182752528)

[Problema 2 23](#_Toc182752529)

[Código: 23](#_Toc182752530)

[Capturas: 26](#_Toc182752531)

[Mata Coronado Orlando Fabián: 26](#_Toc182752532)

[Melendez Aguilera Daniela: 27](#_Toc182752533)

[Problema 3 28](#_Toc182752534)

[Código: 28](#_Toc182752535)

[Capturas 31](#_Toc182752536)

[Mata Coronado Orlando: 31](#_Toc182752537)

[Melendez Aguilera Daniela: 32](#_Toc182752538)

[Problema 4 33](#_Toc182752539)

[Código: 33](#_Toc182752540)

[Capturas: 35](#_Toc182752541)

[Mata Coronado Orlando Fabian: 35](#_Toc182752542)

[Meléndez Aguilera Daniela: 36](#_Toc182752543)

[Problema 5 37](#_Toc182752544)

[Investigación sobre las excepciones: 37](#_Toc182752545)

[¿Qué es? 37](#_Toc182752546)

[Sintaxis: 37](#_Toc182752547)

[Ejemplo: 38](#_Toc182752548)

[Código: 39](#_Toc182752549)

[Capturas: 42](#_Toc182752550)

[Mata Coronado Orlando Fabián: 42](#_Toc182752551)

[Melendez Aguilera Daniela: 43](#_Toc182752552)

[Conclusiones individuales 44](#_Toc182752553)

[Orlando Mata Coronado: 44](#_Toc182752554)

[Daniela Meléndez Aguilera: 44](#_Toc182752555)

[Conclusion general en ingles 45](#_Toc182752556)

[Conclusión general en español 46](#_Toc182752557)

[Valores de la UANL 47](#_Toc182752558)

[Bibliografía 48](#_Toc182752559)

# Introducción (ANECA 2.1)

La programación en PL/SQL y su uso de estructuras de datos avanzadas y objetos complementarios, es importante para la resolución de problemas complejos en casos de negocios. Estos enfoques nos permiten diseñar soluciones escalables, eficientes y organizadas aprovechando las capacidades de las bases de datos para manejar grandes volúmenes de datos de manera inteligente.

En este trabajo se presenta un problema que desglosa varios escenarios que requieren el diseño y desarrollo de diferentes soluciones utilizando la estructura de PL/SQL como procedimientos y funciones. Estas herramientas juntas con algunas otras vistas en el semestre ayudaran a resolver los casos de los diferentes escenarios que se nos presentan que así mismo reflejan problemas que pueden llegar a ser reales en la gestión y análisis de datos.

El objetivo principal de este trabajo es desarrollar soluciones para los ejercicios presentados en PL/SQL en donde se utilicen funciones, excepción, procedimientos. Para abordar las necesidades se proponen el desarrollo de los sistemas de control para los requerimientos que permita gestionar los diferentes datos dados.

Las solucione planteadas buscan resolver el caso planteado haciendo el uso correcto de las diferentes herramientas de PL/SQL aprendidas a lo largo del semestre.

# Código DDL

## Código:

CREATE TABLE GRUPO\_ARTISTICO(

idGrupo NUMBER PRIMARY KEY NOT NULL,

nombre\_grupo VARCHAR(200) NOT NULL

);

CREATE TABLE EVENTO(

idEvento NUMBER PRIMARY KEY,

fecha\_evento DATE NOT NULL,

tipo\_evento VARCHAR(200) NOT NULL,

idGrupo NUMBER NOT NULL,

FOREIGN KEY (idGrupo) REFERENCES GRUPO\_ARTISTICO(idGrupo)

);

CREATE TABLE SEDE(

idSede NUMBER PRIMARY KEY NOT NULL,

nombre\_sede VARCHAR(200) NOT NULL,

direccion\_sede VARCHAR(200) NOT NULL

);

CREATE TABLE PATROCINADOR(

idPatrocinador NUMBER PRIMARY KEY NOT NULL,

nombre\_patrocinador VARCHAR(200) NOT NULL,

monto\_patrocinado FLOAT NOT NULL

);

CREATE TABLE EVENTO\_PATROCINIO(

idEvento NUMBER NOT NULL,

idPatrocinador NUMBER NOT NULL,

monto\_patrocinado FLOAT NOT NULL,

FOREIGN KEY (idEvento) REFERENCES EVENTO (idEvento),

FOREIGN KEY (idPatrocinador) REFERENCES PATROCINADOR (idPatrocinador)

);

--INSERTS DE LOS DATOS A TRABAJAR

-- Tabla Grupo\_Artistico

INSERT INTO Grupo\_Artistico (idGrupo, nombre\_grupo) VALUES (1, 'Los Melódicos');

INSERT INTO Grupo\_Artistico (idGrupo, nombre\_grupo) VALUES (2, 'Armonía Total');

INSERT INTO Grupo\_Artistico (idGrupo, nombre\_grupo) VALUES (3, 'Ritmo y Sabor');

INSERT INTO Grupo\_Artistico (idGrupo, nombre\_grupo) VALUES (4, 'Voces Unidas');

INSERT INTO Grupo\_Artistico (idGrupo, nombre\_grupo) VALUES (5, 'Sonido Urbano');

-- Tabla Evento

INSERT INTO Evento (idEvento, fecha\_evento, tipo\_evento, idGrupo) VALUES (1, '11-20-2024', 'Local', 1);

INSERT INTO Evento (idEvento, fecha\_evento, tipo\_evento, idGrupo) VALUES (2, '11-22-2024', 'Foráneo', 2);

INSERT INTO Evento (idEvento, fecha\_evento, tipo\_evento, idGrupo) VALUES (3, '11-25-2024', 'Local', 3);

INSERT INTO Evento (idEvento, fecha\_evento, tipo\_evento, idGrupo) VALUES (4, '11-27-2024', 'Foráneo', 4);

INSERT INTO Evento (idEvento, fecha\_evento, tipo\_evento, idGrupo) VALUES (5, '11-30-2024', 'Local', 5);

-- Tabla Sede

INSERT INTO Sede (idSede, nombre\_sede, direccion\_sede) VALUES (1, 'Auditorio Central', 'Calle Principal #123');

INSERT INTO Sede (idSede, nombre\_sede, direccion\_sede) VALUES (2, 'Plaza Cultural', 'Av. Las Rosas #45');

INSERT INTO Sede (idSede, nombre\_sede, direccion\_sede) VALUES (3, 'Teatro Nacional', 'Paseo de la Reforma #890');

INSERT INTO Sede (idSede, nombre\_sede, direccion\_sede) VALUES (4, 'Parque del Sol', 'Av. del Parque #33');

INSERT INTO Sede (idSede, nombre\_sede, direccion\_sede) VALUES (5, 'Estadio Urbano', 'Calle Independencia #12');

-- Tabla Patrocinador

INSERT INTO Patrocinador (idPatrocinador, nombre\_patrocinador, monto\_patrocinado) VALUES (1, 'Empresa A', 5000);

INSERT INTO Patrocinador (idPatrocinador, nombre\_patrocinador, monto\_patrocinado) VALUES (2, 'Empresa B', 8000);

INSERT INTO Patrocinador (idPatrocinador, nombre\_patrocinador, monto\_patrocinado) VALUES (3, 'Fundación C', 10000);

INSERT INTO Patrocinador (idPatrocinador, nombre\_patrocinador, monto\_patrocinado) VALUES (4, 'Comercio D', 3000);

INSERT INTO Patrocinador (idPatrocinador, nombre\_patrocinador, monto\_patrocinado) VALUES (5, 'Organización E', 7000);

-- Tabla Evento\_Patrocinio

INSERT INTO Evento\_Patrocinio (idEvento, idPatrocinador, monto\_patrocinado) VALUES (2, 1, 5000);

INSERT INTO Evento\_Patrocinio (idEvento, idPatrocinador, monto\_patrocinado) VALUES (2, 3, 8000);

INSERT INTO Evento\_Patrocinio (idEvento, idPatrocinador, monto\_patrocinado) VALUES (4, 2, 10000);

INSERT INTO Evento\_Patrocinio (idEvento, idPatrocinador, monto\_patrocinado) VALUES (4, 4, 3000);

INSERT INTO Evento\_Patrocinio (idEvento, idPatrocinador, monto\_patrocinado) VALUES (4, 5, 7000);

# Problema 1

## Código:

--PROBLEMA 1 - CRUD

CREATE OR REPLACE PROCEDURE sp\_crud\_evento (

accion IN VARCHAR2,

pidEvento IN EVENTO.idEvento%TYPE,

pfecha\_evento IN EVENTO.fecha\_evento%TYPE,

ptipo\_evento IN EVENTO.tipo\_evento%TYPE,

pidGrupo IN EVENTO.idGrupo%TYPE,

mensaje OUT VARCHAR2

)

AS

existe NUMBER;

empalme NUMBER;

BEGIN

IF accion NOT IN ('INSERT', 'UPDATE', 'DELETE', 'READ') THEN

mensaje := 'Acción no válida. Use INSERT, UPDATE, DELETE o READ.';

RETURN;

END IF;

IF accion IN ('INSERT', 'UPDATE') THEN

SELECT COUNT(\*) INTO existe

FROM evento

WHERE idEvento = pidEvento;

SELECT COUNT(\*) INTO empalme

FROM evento

WHERE idGrupo = pidGrupo

AND fecha\_evento = pfecha\_evento

AND (accion = 'UPDATE' OR idEvento != pidEvento);

END IF;

IF accion = 'INSERT' THEN

IF existe > 0 THEN

mensaje := 'Error: El idEvento ya existe.';

ELSIF empalme > 0 THEN

mensaje := 'Error: El evento se empalma con otro evento del mismo grupo.';

ELSE

INSERT INTO evento (idEvento, fecha\_evento, tipo\_evento, idGrupo)

VALUES (pidEvento, pfecha\_evento, ptipo\_evento, pidGrupo);

mensaje := 'Evento insertado correctamente.';

END IF;

ELSIF accion = 'UPDATE' THEN

IF existe = 0 THEN

mensaje := 'Error: El idEvento no existe.';

ELSIF empalme > 0 THEN

mensaje := 'Error: El evento se empalma con otro evento del mismo grupo.';

ELSE

UPDATE evento

SET fecha\_evento = pfecha\_evento, tipo\_evento = ptipo\_evento, idGrupo = pidGrupo

WHERE idEvento = pidEvento;

mensaje := 'Evento actualizado correctamente.';

END IF;

ELSIF accion = 'DELETE' THEN

IF existe = 0 THEN

mensaje := 'Error: El idEvento no existe.';

ELSE

DELETE FROM evento WHERE idEvento = pidEvento;

mensaje := 'Evento eliminado correctamente.';

END IF;

ELSIF accion = 'READ' THEN

mensaje := 'Consulta realizada.';

END IF;

END;

--PRUEBA DEL CRUD PROBLEMA 1

--READ

DECLARE

v\_mensaje VARCHAR2(100);

BEGIN

sp\_crud\_evento(

accion => 'DELETE',

pidEvento => 1,

pfecha\_evento => NULL,

ptipo\_evento => NULL,

pidGrupo => NULL,

mensaje => v\_mensaje

);

DBMS\_OUTPUT.PUT\_LINE(v\_mensaje);

END;

--INSERT

DECLARE

v\_mensaje VARCHAR2(100);

BEGIN

sp\_crud\_evento(

accion => 'INSERT',

pidEvento => 1,

pfecha\_evento => TO\_DATE('2024-12-25', 'YYYY-MM-DD'),

ptipo\_evento => 'Fiesta',

pidGrupo => 1,

mensaje => v\_mensaje

);

DBMS\_OUTPUT.PUT\_LINE(v\_mensaje);

END;

--UPDATE

DECLARE

v\_mensaje VARCHAR2(100);

BEGIN

sp\_crud\_evento(

accion => 'UPDATE',

pidEvento => 1,

pfecha\_evento => TO\_DATE('2025-01-01', 'YYYY-MM-DD'),

ptipo\_evento => 'Reunión',

pidGrupo => 2,

mensaje => v\_mensaje

);

DBMS\_OUTPUT.PUT\_LINE(v\_mensaje);

END;

--DELETE

DECLARE

v\_mensaje VARCHAR2(100);

BEGIN

sp\_crud\_evento(

accion => 'DELETE',

pidEvento => 1,

pfecha\_evento => NULL,

ptipo\_evento => NULL,

pidGrupo => NULL,

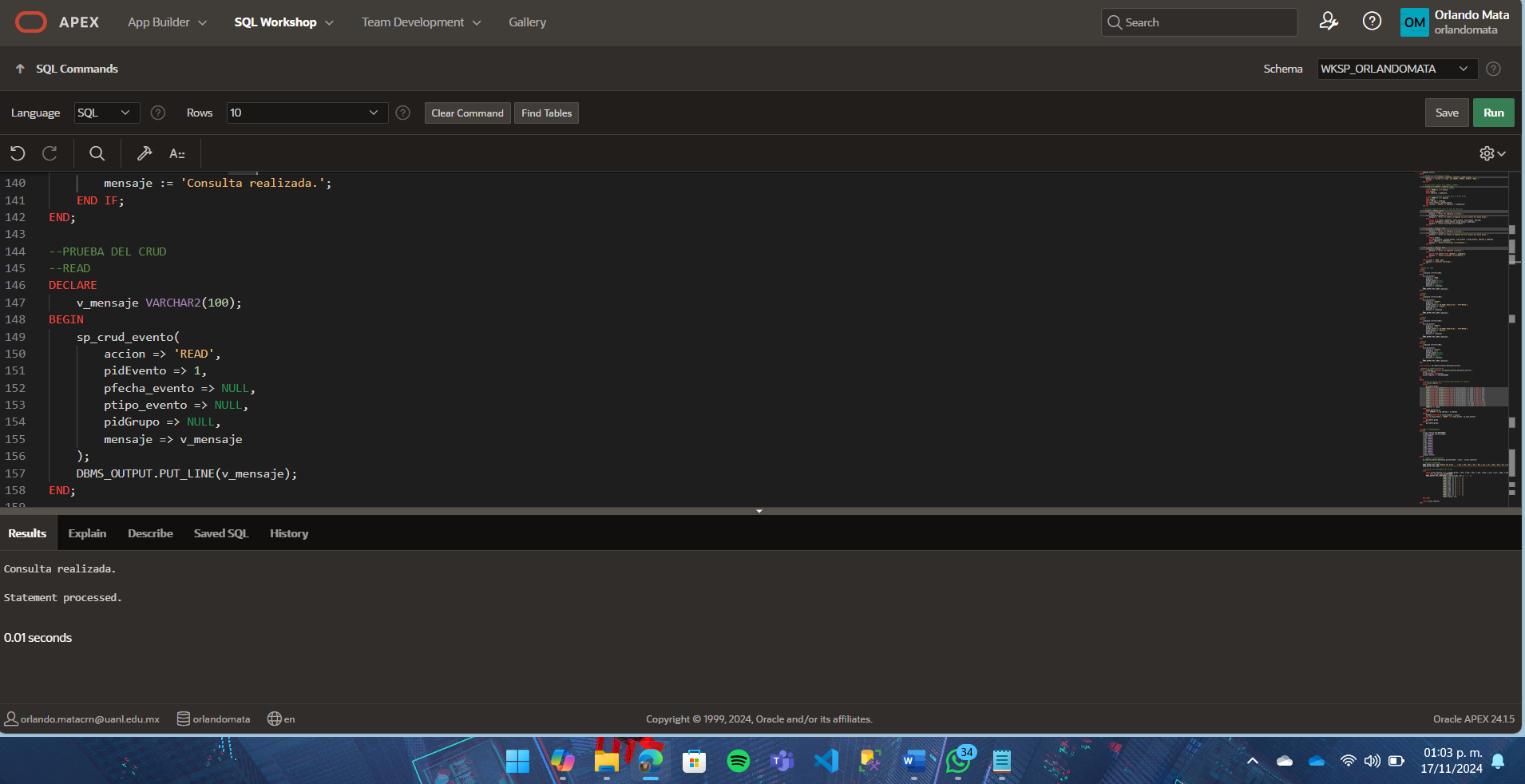
mensaje => v\_mensaje

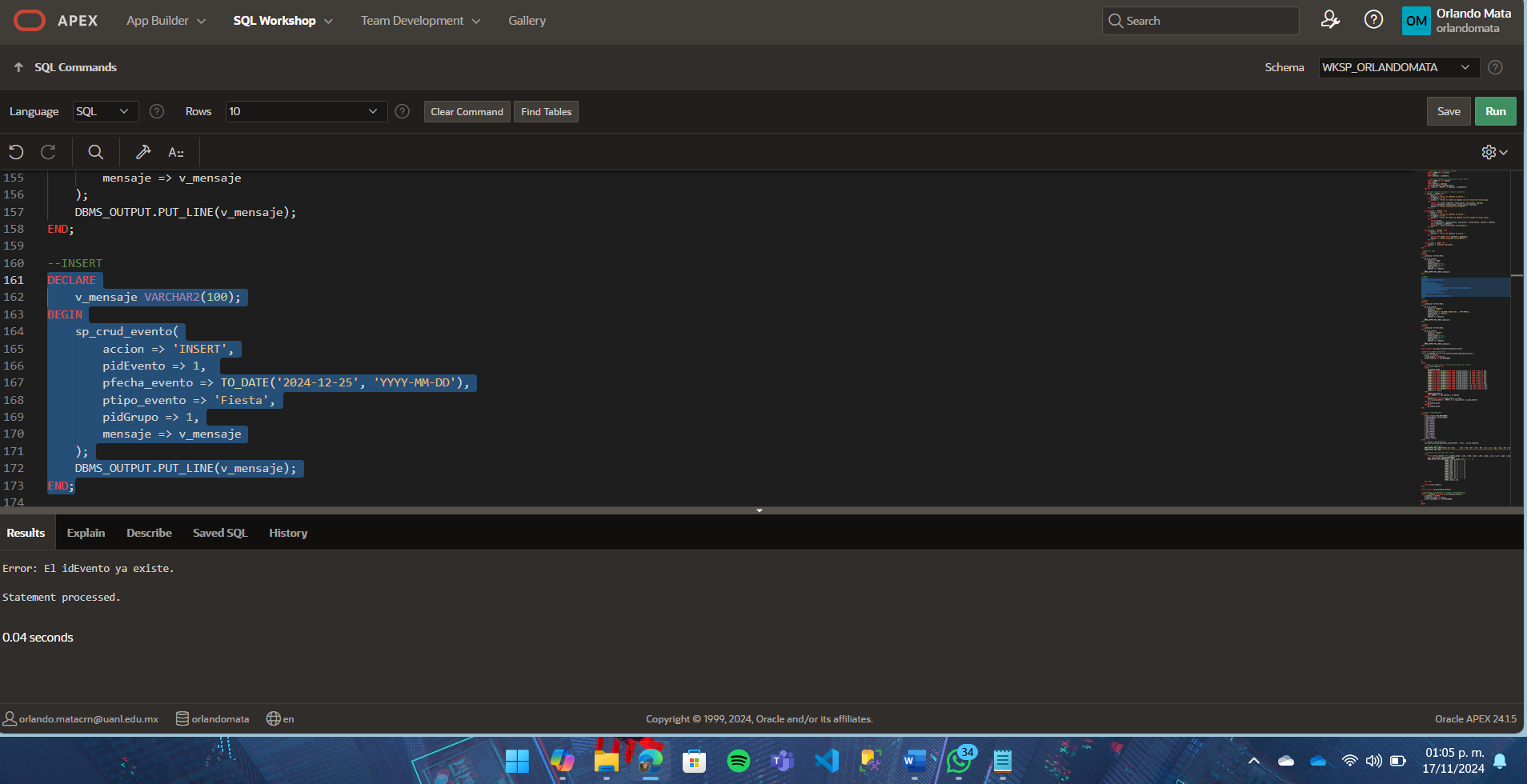
);

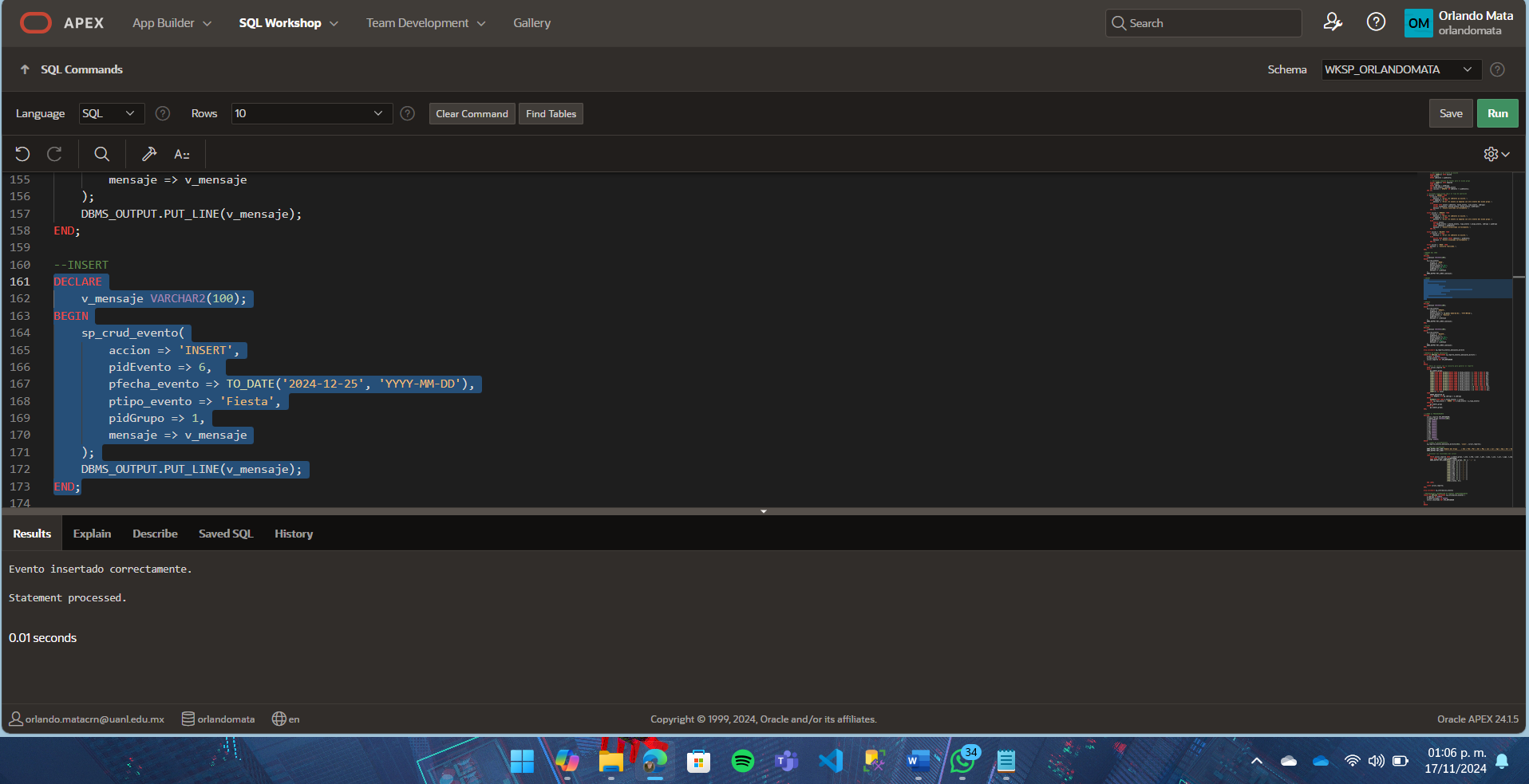
DBMS\_OUTPUT.PUT\_LINE(v\_mensaje);

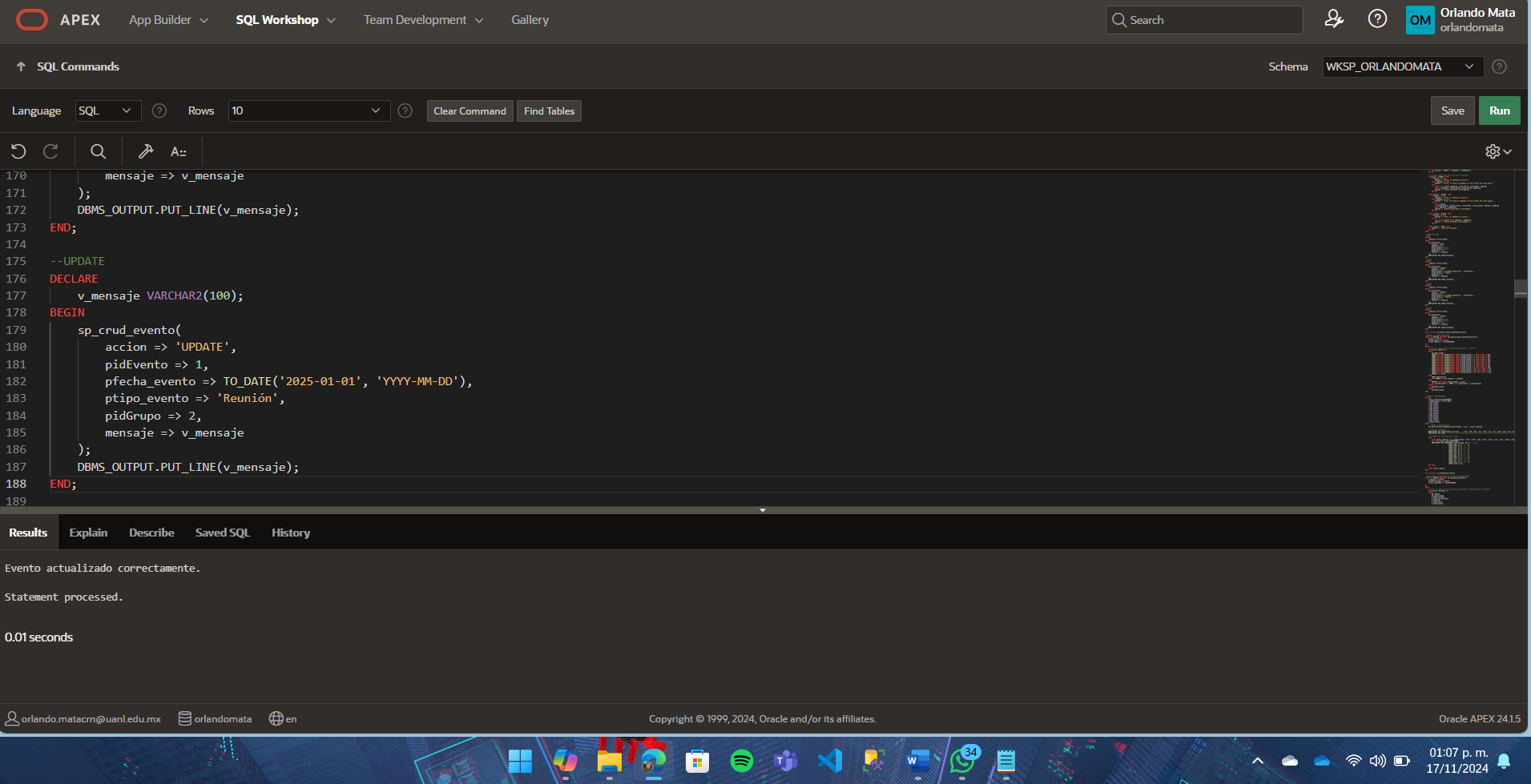
END;

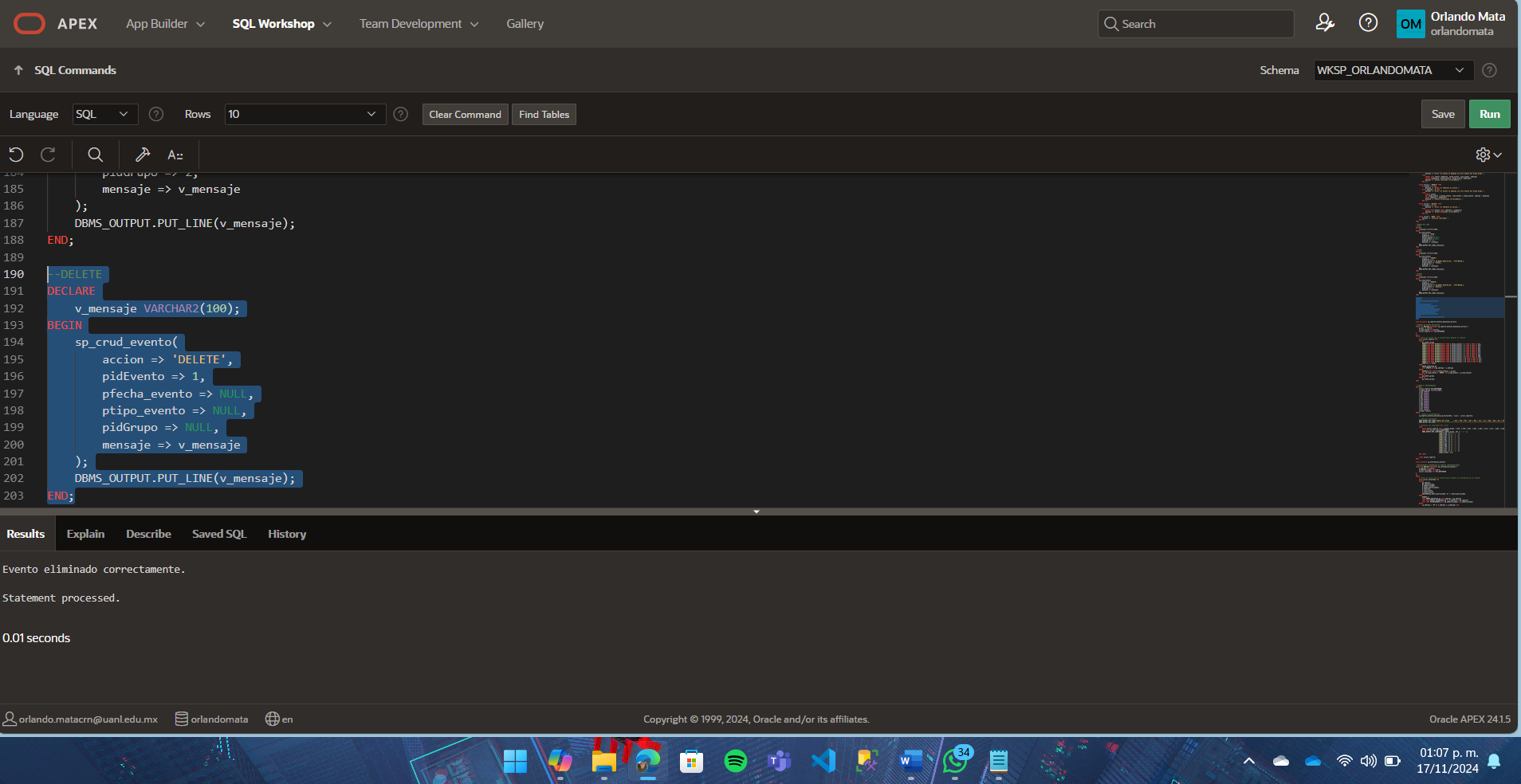
## Capturas:

Mata Coronado Orlando Fabián:









### Melendez Aguilera Daniela:

Captura de pantalla de computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

# Problema 2

## Código:

--REPORTE DE GRUPOS ARTISTICOS

CREATE OR REPLACE PROCEDURE sp\_reporte\_eventos\_mensuales\_directo (

p\_anio IN NUMBER,

p\_tipo\_evento IN VARCHAR2,

cursor\_reporte OUT SYS\_REFCURSOR

)

IS

BEGIN

OPEN cursor\_reporte FOR

SELECT

ga.nombre\_grupo,

COUNT(CASE WHEN EXTRACT(MONTH FROM e.fecha\_evento) = 1 THEN 1 END) AS ene,

COUNT(CASE WHEN EXTRACT(MONTH FROM e.fecha\_evento) = 2 THEN 1 END) AS feb,

COUNT(CASE WHEN EXTRACT(MONTH FROM e.fecha\_evento) = 3 THEN 1 END) AS mar,

COUNT(CASE WHEN EXTRACT(MONTH FROM e.fecha\_evento) = 4 THEN 1 END) AS abr,

COUNT(CASE WHEN EXTRACT(MONTH FROM e.fecha\_evento) = 5 THEN 1 END) AS may,

COUNT(CASE WHEN EXTRACT(MONTH FROM e.fecha\_evento) = 6 THEN 1 END) AS jun,

COUNT(CASE WHEN EXTRACT(MONTH FROM e.fecha\_evento) = 7 THEN 1 END) AS jul,

COUNT(CASE WHEN EXTRACT(MONTH FROM e.fecha\_evento) = 8 THEN 1 END) AS ago,

COUNT(CASE WHEN EXTRACT(MONTH FROM e.fecha\_evento) = 9 THEN 1 END) AS sep,

COUNT(CASE WHEN EXTRACT(MONTH FROM e.fecha\_evento) = 10 THEN 1 END) AS oct,

COUNT(CASE WHEN EXTRACT(MONTH FROM e.fecha\_evento) = 11 THEN 1 END) AS nov,

COUNT(CASE WHEN EXTRACT(MONTH FROM e.fecha\_evento) = 12 THEN 1 END) AS dic,

COUNT(\*) AS total

FROM

GRUPO\_ARTISTICO ga

JOIN EVENTO e ON ga.idGrupo = e.idGrupo

WHERE

EXTRACT(YEAR FROM e.fecha\_evento) = p\_anio

AND (p\_tipo\_evento = 'AMBOS' OR e.tipo\_evento = p\_tipo\_evento)

GROUP BY

ga.nombre\_grupo

ORDER BY

ga.nombre\_grupo;

END;

--LLAMAR AL PROCEDIMIENTO

DECLARE

cursor\_reporte SYS\_REFCURSOR;

v\_nombre\_grupo VARCHAR2(200);

v\_ene NUMBER;

v\_feb NUMBER;

v\_mar NUMBER;

v\_abr NUMBER;

v\_may NUMBER;

v\_jun NUMBER;

v\_jul NUMBER;

v\_ago NUMBER;

v\_sep NUMBER;

v\_oct NUMBER;

v\_nov NUMBER;

v\_dic NUMBER;

v\_total NUMBER;

BEGIN

sp\_reporte\_eventos\_mensuales\_directo(2024, 'Local', cursor\_reporte);

DBMS\_OUTPUT.PUT\_LINE('Nombre del Grupo | Ene | Feb | Mar | Abr | May | Jun | Jul | Ago | Sep | Oct | Nov | Dic | Total');

DBMS\_OUTPUT.PUT\_LINE('---------------------------------------------------------------------------------------------');

LOOP

FETCH cursor\_reporte INTO v\_nombre\_grupo, v\_ene, v\_feb, v\_mar, v\_abr, v\_may, v\_jun, v\_jul, v\_ago, v\_sep, v\_oct, v\_nov, v\_dic, v\_total;

EXIT WHEN cursor\_reporte%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(RPAD(v\_nombre\_grupo, 20) || ' | ' ||

LPAD(v\_ene, 3) || ' | ' ||

LPAD(v\_feb, 3) || ' | ' ||

LPAD(v\_mar, 3) || ' | ' ||

LPAD(v\_abr, 3) || ' | ' ||

LPAD(v\_may, 3) || ' | ' ||

LPAD(v\_jun, 3) || ' | ' ||

LPAD(v\_jul, 3) || ' | ' ||

LPAD(v\_ago, 3) || ' | ' ||

LPAD(v\_sep, 3) || ' | ' ||

LPAD(v\_oct, 3) || ' | ' ||

LPAD(v\_nov, 3) || ' | ' ||

LPAD(v\_dic, 3) || ' | ' ||

LPAD(v\_total, 5));

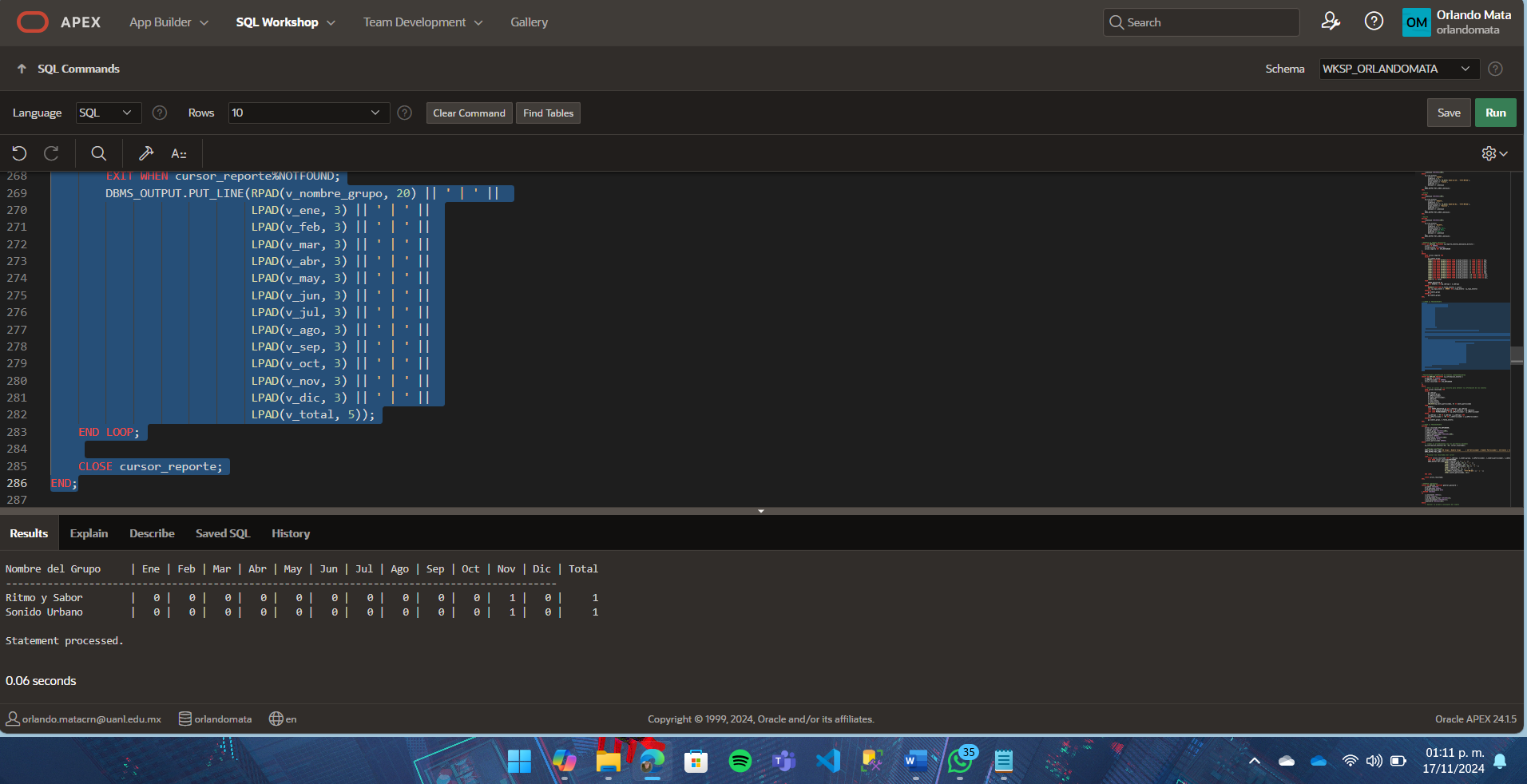
END LOOP;

CLOSE cursor\_reporte;

END;

## Capturas:

### Mata Coronado Orlando Fabián:



### Melendez Aguilera Daniela:

Captura de pantalla de computadora

Descripción generada automáticamente

# Problema 3

## Código:

CREATE OR REPLACE PROCEDURE sp\_informacion\_eventos (

p\_idGrupo IN NUMBER,

p\_idPatrocinador IN NUMBER,

cursor\_resultado OUT SYS\_REFCURSOR

)

IS

BEGIN

OPEN cursor\_resultado FOR

SELECT

ga.idGrupo,

ga.nombre\_grupo,

p.idPatrocinador,

p.nombre\_patrocinador,

e.idEvento,

e.tipo\_evento,

e.fecha\_evento,

COALESCE(ep.monto\_patrocinado, 0) AS monto\_patrocinado

FROM

EVENTO e

JOIN GRUPO\_ARTISTICO ga ON e.idGrupo = ga.idGrupo

LEFT JOIN EVENTO\_PATROCINIO ep ON e.idEvento = ep.idEvento

LEFT JOIN PATROCINADOR p ON ep.idPatrocinador = p.idPatrocinador

WHERE

(p\_idGrupo = -99 OR e.idGrupo = p\_idGrupo) AND

(p\_idPatrocinador = -99 OR p.idPatrocinador = p\_idPatrocinador)

ORDER BY

ga.nombre\_grupo, e.fecha\_evento;

END;

--LLAMAR AL PROCEDIMIENTO

DECLARE

cursor\_resultado SYS\_REFCURSOR;

v\_idGrupo NUMBER;

v\_nombre\_grupo VARCHAR2(200);

v\_idPatrocinador NUMBER;

v\_nombre\_patrocinador VARCHAR2(200);

v\_idEvento NUMBER;

v\_tipo\_evento VARCHAR2(200);

v\_fecha\_evento DATE;

v\_monto\_patrocinado NUMBER;

BEGIN

sp\_informacion\_eventos(-99, -99, cursor\_resultado);

DBMS\_OUTPUT.PUT\_LINE('ID Grupo | Nombre Grupo | ID Patrocinador | Nombre Patrocinador | ID Evento | Tipo Evento | Fecha Evento | Monto Patrocinado');

DBMS\_OUTPUT.PUT\_LINE('---------------------------------------------------------------------------------------------------------');

LOOP

FETCH cursor\_resultado INTO v\_idGrupo, v\_nombre\_grupo, v\_idPatrocinador, v\_nombre\_patrocinador, v\_idEvento, v\_tipo\_evento, v\_fecha\_evento, v\_monto\_patrocinado;

EXIT WHEN cursor\_resultado%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(LPAD(v\_idGrupo, 8) || ' | ' ||

RPAD(v\_nombre\_grupo, 18) || ' | ' ||

LPAD(v\_idPatrocinador, 15) || ' | ' ||

RPAD(v\_nombre\_patrocinador, 20) || ' | ' ||

LPAD(v\_idEvento, 9) || ' | ' ||

RPAD(v\_tipo\_evento, 12) || ' | ' ||

TO\_CHAR(v\_fecha\_evento, 'YYYY-MM-DD') || ' | ' ||

LPAD(v\_monto\_patrocinado, 16));

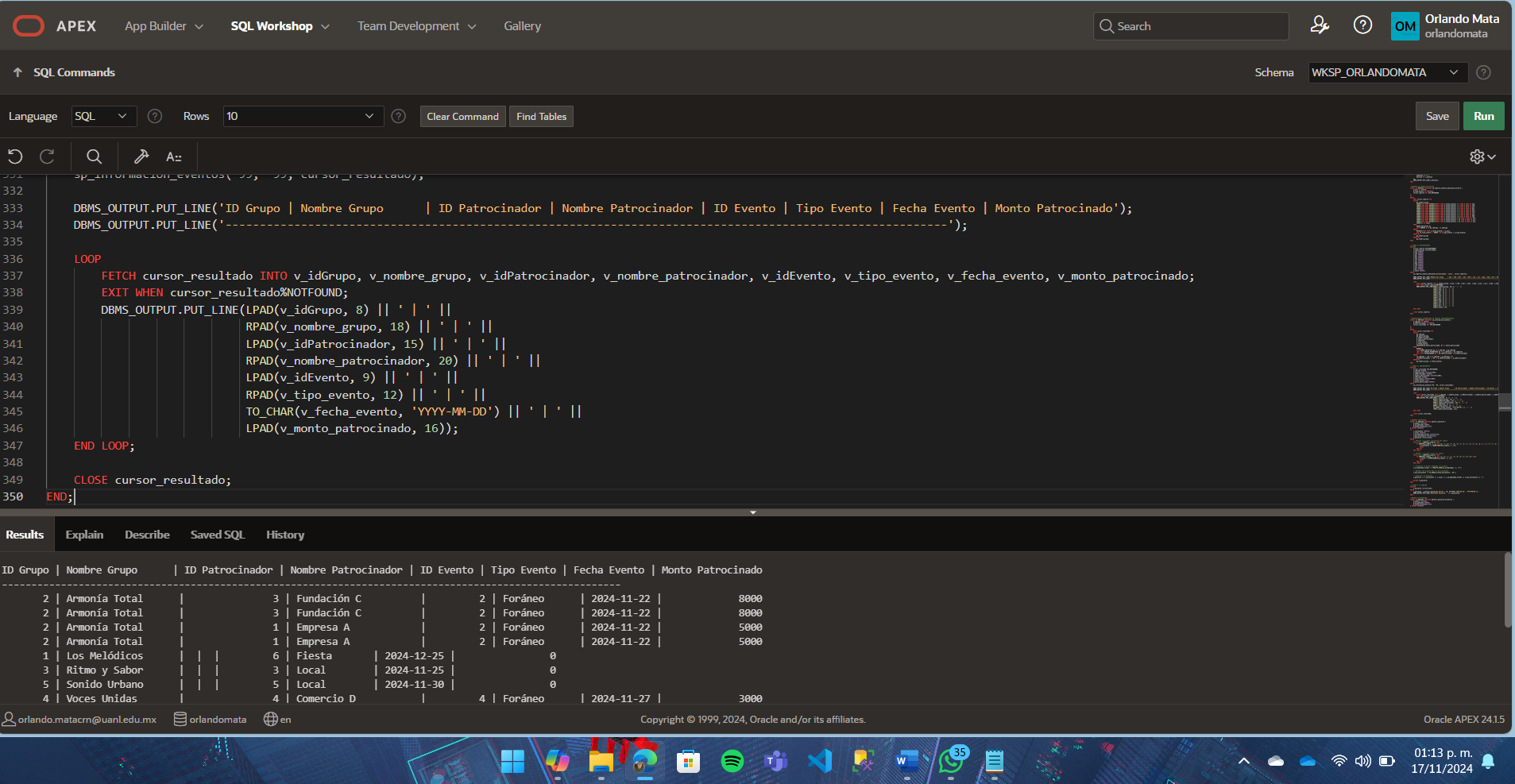
END LOOP;

CLOSE cursor\_resultado;

END;

## Capturas

### Mata Coronado Orlando:



### Melendez Aguilera Daniela:

Captura de pantalla de computadora

Descripción generada automáticamente

# Problema 4

## Código:

CREATE OR REPLACE FUNCTION generar\_password (

p\_nombre VARCHAR2,

p\_id\_empleado NUMBER,

p\_fecha\_nacimiento DATE

) RETURN VARCHAR2

IS

v\_consonante CHAR(1);

v\_vocal CHAR(1);

v\_id\_empleado\_format VARCHAR2(5);

v\_dia\_nacimiento VARCHAR2(2);

v\_password VARCHAR2(50);

BEGIN

FOR i IN 1..LENGTH(p\_nombre) LOOP

IF SUBSTR(p\_nombre, i, 1) IN ('b','c','d','f','g','h','j','k','l','m','n','p','q','r','s','t','v','w','x','y','z','B','C','D','F','G','H','J','K','L','M','N','P','Q','R','S','T','V','W','X','Y','Z') THEN

v\_consonante := LOWER(SUBSTR(p\_nombre, i, 1));

EXIT;

END IF;

END LOOP;

FOR i IN 1..LENGTH(p\_nombre) LOOP

IF SUBSTR(p\_nombre, i, 1) IN ('a','e','i','o','u','A','E','I','O','U') THEN

v\_vocal := UPPER(SUBSTR(p\_nombre, i, 1));

EXIT;

END IF;

END LOOP;

v\_id\_empleado\_format := LPAD(TO\_CHAR(p\_id\_empleado), 5, '0');

v\_dia\_nacimiento := TO\_CHAR(p\_fecha\_nacimiento, 'DD');

v\_password := v\_consonante || v\_vocal || v\_id\_empleado\_format || v\_dia\_nacimiento || '\*';

RETURN v\_password;

END;

--LLAMAR A LA FUNCION

DECLARE

v\_password VARCHAR2(50);

BEGIN

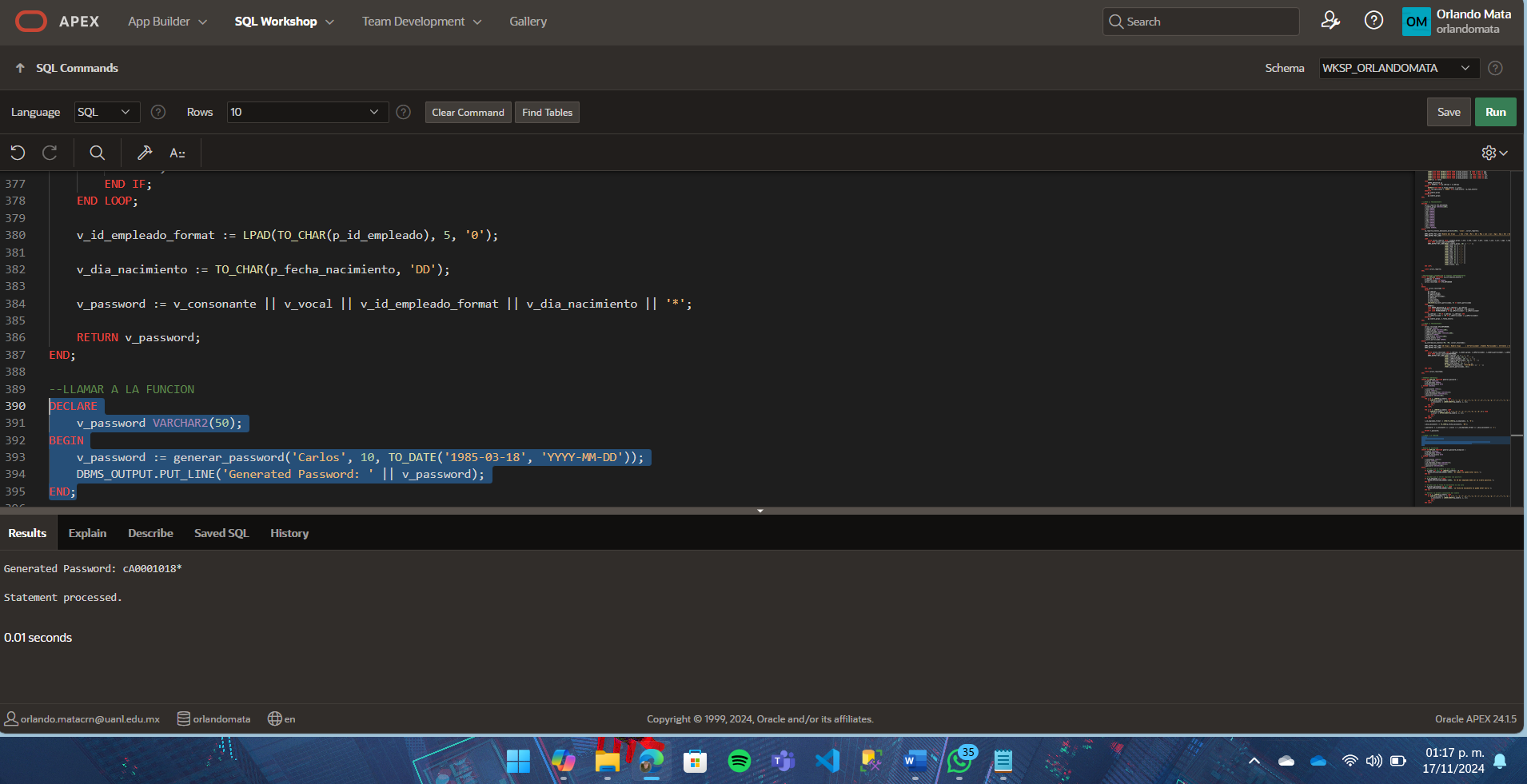
v\_password := generar\_password('Carlos', 10, TO\_DATE('1985-03-18', 'YYYY-MM-DD'));

DBMS\_OUTPUT.PUT\_LINE('Generated Password: ' || v\_password);

END;

## Capturas:

### Mata Coronado Orlando Fabian:



### Meléndez Aguilera Daniela:

Captura de pantalla de computadora

Descripción generada automáticamente

# Problema 5

## Investigación sobre las excepciones:

### ¿Qué es?

Una excepción es una situación especial dentro de la ejecución de un programa, que puede ser capturada para asignar un nuevo comportamiento. Una excepción puede ser un error de ejecución (una división entre 0) o cualquier otro tipo de suceso.

Las excepciones deben ser declaradas dentro de la sección DECLARE, como si de una variable se tratase:

**DECLARE**

e\_sin\_alumnos **EXCEPTION**;

Una vez que la excepción está definida, ésta debe ser lanzada, ya sea automáticamente por Oracle (cuando se produce un error controlado por Oracle), o lanzada manualmente por el usuario a través de la instrucción **RAISE <excepcion>.**

### Sintaxis:

**EXCEPTION**

**WHEN** nb\_excepcion\_1 **THEN**

instrucciones excep1;

**WHEN** nb\_excepcion\_2 **THEN**

instrucciones excep2;

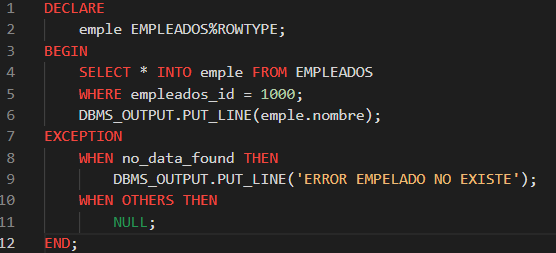
**WHEN** nb\_excepcion\_n **THEN**

instrucciones excepn;

[**WHEN** OTHERS **THEN**

instrucciones;]

### Ejemplo:



## Código:

CREATE OR REPLACE FUNCTION generar\_password\_excepcion (

p\_nombre VARCHAR2,

p\_id\_empleado NUMBER,

p\_fecha\_nacimiento DATE

) RETURN VARCHAR2

IS

v\_consonante CHAR(1);

v\_vocal CHAR(1);

v\_id\_empleado\_format VARCHAR2(5);

v\_dia\_nacimiento VARCHAR2(2);

v\_password VARCHAR2(50);

BEGIN

IF p\_nombre IS NULL OR LENGTH(p\_nombre) = 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'El nombre no puede estar vacío.');

END IF;

IF p\_id\_empleado <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'El ID del empleado debe ser un número positivo.');

END IF;

IF p\_fecha\_nacimiento IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20003, 'La fecha de nacimiento no puede estar vacía.');

END IF;

FOR i IN 1..LENGTH(p\_nombre) LOOP

IF SUBSTR(p\_nombre, i, 1) IN ('b','c','d','f','g','h','j','k','l','m','n','p','q','r','s','t','v','w','x','y','z','B','C','D','F','G','H','J','K','L','M','N','P','Q','R','S','T','V','W','X','Y','Z') THEN

v\_consonante := LOWER(SUBSTR(p\_nombre, i, 1));

EXIT;

END IF;

END LOOP;

FOR i IN 1..LENGTH(p\_nombre) LOOP

IF SUBSTR(p\_nombre, i, 1) IN ('a','e','i','o','u','A','E','I','O','U') THEN

v\_vocal := UPPER(SUBSTR(p\_nombre, i, 1));

EXIT;

END IF;

END LOOP;

IF v\_consonante IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20004, 'No se encontró ninguna consonante en el nombre.');

END IF;

IF v\_vocal IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20005, 'No se encontró ninguna vocal en el nombre.');

END IF;

v\_id\_empleado\_format := LPAD(TO\_CHAR(p\_id\_empleado), 5, '0');

v\_dia\_nacimiento := TO\_CHAR(p\_fecha\_nacimiento, 'DD');

v\_password := v\_consonante || v\_vocal || v\_id\_empleado\_format || v\_dia\_nacimiento || '\*';

RETURN v\_password;

EXCEPTION

WHEN OTHERS THEN

RETURN 'Error: ' || SQLERRM;

END;

--LLAMAR FUNCION DE EXCEPCION

DECLARE

v\_password VARCHAR2(50);

BEGIN

v\_password := generar\_password('Julian', 12, TO\_DATE('1985-03-18', 'YYYY-MM-DD'));

DBMS\_OUTPUT.PUT\_LINE('Generated Password: ' || v\_password);

EXCEPTION

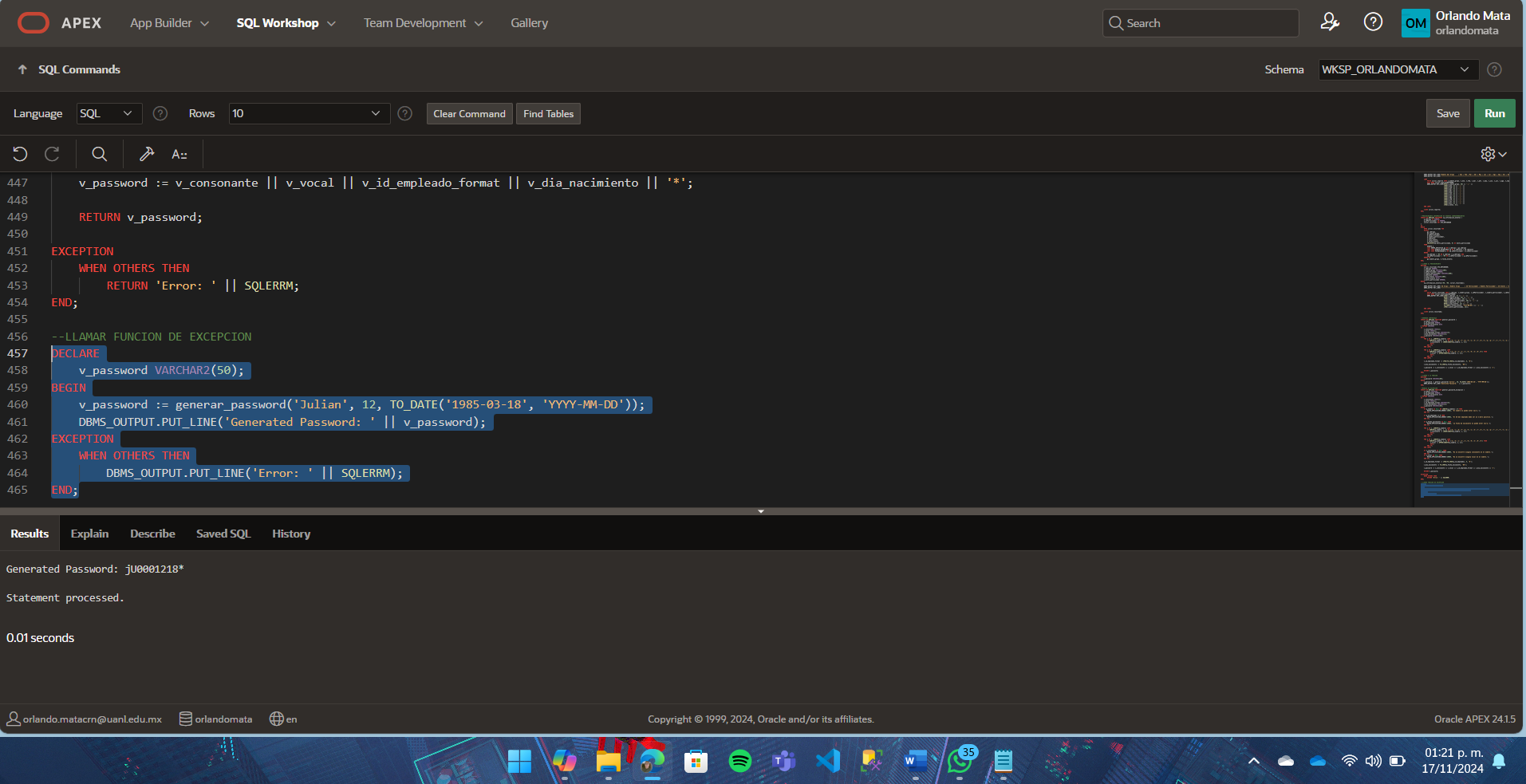
WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

## Capturas:

### Mata Coronado Orlando Fabián:



### Melendez Aguilera Daniela:

Captura de pantalla de computadora

Descripción generada automáticamente

# Conclusiones individuales

## Orlando Mata Coronado:

En conclusión, resulto muy interesante la toma de esta materia ya que hasta el momento el único lenguaje que se nos mostro fue Python y descubrir las posibilidades que PL/SQL te brinda y la manera distinta de programar a comparación de Python hace que sea un lenguaje interesante y un poco complicado de comprender ya que no se esta acostumbrado a un lenguaje que funciona mediante bloques de código bloques.

El desarrollo de nuestro documento trata del desarrollo de un sistema que permita gestionar los eventos culturales de nuestra facultad, fue interesante el desarrollo de este ya que el hecho que sea un programa que la universidad esta solicitando te hace sentir que perteneces a esta facultad aparte durante el desarrollo también sirvió como un repaso a la hora de practicar código aprendido en clase, como los procedimientos y las variables de salida ya que por mucho fue en lo personal lo que mas se me dificulto aprender pero gracias a esta evidencia me queda más claro su funcionamiento así como el de las funciones y los bloques anónimos.

## Daniela Meléndez Aguilera:

En resumen, la creación de código en PL/SQL posibilita la resolución de desafíos complejos en la administración y evaluación de información a través de funciones, procedimientos y control de errores. Este trabajo personal utiliza estas herramientas para crear soluciones prácticas y organizadas, evidenciando la efectividad de aplicar los conceptos aprendidos en situaciones empresariales reales, fortaleciendo el aprendizaje y su uso en el ámbito profesional.

# Conclusion general en ingles

In conclusion, programming in PL/SQL, through the use of advanced data structures and tools such as functions, procedures and exception handling, is a powerful tool to solve complex problems in business contexts. This approach allows us to develop robust, scalable and organized solutions, taking advantage of the capabilities of databases to process and manage large volumes of information efficiently.

The work developed puts these concepts into practice, facing scenarios that simulate real problems in data management. Each case is solved with a methodical design and the appropriate use of the tools learned, demonstrating how PL/SQL can be applied to structure and automate business processes with precision and efficiency.

In this way, the development of the proposed solutions not only meets the academic objectives, but also reflects practical applications that can be used in real situations, strengthening the link between theoretical learning and its implementation in the professional world.

# Conclusión general en español

En resumen, el uso de estructuras de datos avanzadas y funciones en PL/SQL, junto con procedimientos y manejo de excepciones, es una herramienta efectiva para abordar problemas complejos en entornos empresariales. Este método posibilita crear respuestas resistentes, adaptables y bien estructuradas, haciendo uso de las capacidades de las bases de datos para manejar de forma eficaz cantidades extensas de datos.

La implementación de estos conceptos en el trabajo implica enfrentar situaciones que imitan problemas reales en la administración de datos. Cada situación se soluciona con un enfoque sistemático y la correcta aplicación de las herramientas adquiridas, mostrando la forma en que PL/SQL puede ser usado para organizar y automatizar con precisión y eficacia las operaciones comerciales.

Así, la creación de las soluciones sugeridas no solo alcanza los objetivos académicos, sino también muestra aplicaciones prácticas para situaciones reales, mejorando la conexión entre la teoría aprendida y su aplicación en el ámbito laboral.

# Valores de la UANL

La Universidad autónoma de Nuevo León enseña diferentes valores a sus alumnos para cumplir en su vida laboral, estudiantil e incluso vida personal.

Los valores que utilizamos de la UANL fueron:

Responsabilidad al entregar el documento em tempo y forma tal y como nos fue solicitado. También aprendimos lo valioso que es la solidaridad al cada uno como integrante del equipo ayudarnos uno al otro cuando tuvimos dificultades al hacer nuestro proyecto y así salir adelante con él. La tolerancia fue también un valor importante ya que el Trabajo em equipo requiere mucha tolerancia, el respeto mutuo no pudo haber faltado debido a que para nuestro equipo el tratarnos con respeto fue esencial para poder terminar nuestro proyecto de manera correcta.

Y sin más mencionar la honestidad también es un valor importante que aplicamos em este Trabajo debido q que todo lo hicimos de manera correcta evitando el plagio a toda costa y esforzándonos para hacerlo.

# Bibliografía

Alcalde, A. (2016, enero 1). *El Baúl del programador*. PL/SQL. Excepciones. https://elbauldelprogramador.com/plsql-excepciones/

UANL. (2024, junio 25). *Valores y atributos UANL - Universidad Autónoma de Nuevo León*. Universidad Autónoma de Nuevo León; UANL - Universidad Autónoma de Nuevo León. https://www.uanl.mx/valores-y-atibutos/