PRACTICUM (18/06/2025)

ST10212542 – AALIYAH ALLIE

# Full SQL Code

--preload code

-- creation of tables and populations

CREATE TABLE EVENT (

EVENT\_ID INT PRIMARY KEY,

EVENT\_NAME VARCHAR(100),

EVENT\_RATE DECIMAL(10,2)

);

CREATE TABLE ARTIST (

ARTIST\_ID VARCHAR(10) PRIMARY KEY,

ARTIST\_NAME VARCHAR(100),

ARTIST\_EMAIL VARCHAR(100)

);

CREATE TABLE BOOKINGS (

BOOKING\_ID INT PRIMARY KEY,

BOOKING\_DATE DATE,

EVENT\_ID INT,

ARTIST\_ID VARCHAR(10),

FOREIGN KEY (EVENT\_ID) REFERENCES EVENT(EVENT\_ID),

FOREIGN KEY (ARTIST\_ID) REFERENCES ARTIST(ARTIST\_ID)

);

INSERT INTO EVENT (EVENT\_ID, EVENT\_NAME, EVENT\_RATE)

VALUES (1001, 'Open Air Comedy Festival', 300.00);

INSERT INTO EVENT (EVENT\_ID, EVENT\_NAME, EVENT\_RATE)

VALUES (1002, 'Mountain Side Music Festival', 280.00);

INSERT INTO EVENT (EVENT\_ID, EVENT\_NAME, EVENT\_RATE)

VALUES (1003, 'Beach Music Festival', 195.00);

INSERT INTO ARTIST (ARTIST\_ID, ARTIST\_NAME, ARTIST\_EMAIL)

VALUES ('A\_101', 'Max Trillion', 'maxt@isat.com');

INSERT INTO ARTIST (ARTIST\_ID, ARTIST\_NAME, ARTIST\_EMAIL)

VALUES ('A\_102', 'Music Mayhem', 'mayhem@ymail.com');

INSERT INTO ARTIST (ARTIST\_ID, ARTIST\_NAME, ARTIST\_EMAIL)

VALUES ('A\_103', 'LOL Man', 'lol@isat.com');

INSERT INTO BOOKINGS (BOOKING\_ID, BOOKING\_DATE, EVENT\_ID, ARTIST\_ID)

VALUES (1, TO\_DATE('2024-07-15', 'YYYY-MM-DD'), 1002, 'A\_101');

INSERT INTO BOOKINGS (BOOKING\_ID, BOOKING\_DATE, EVENT\_ID, ARTIST\_ID)

VALUES (2, TO\_DATE('2024-07-15', 'YYYY-MM-DD'), 1002, 'A\_102');

INSERT INTO BOOKINGS (BOOKING\_ID, BOOKING\_DATE, EVENT\_ID, ARTIST\_ID)

VALUES (3, TO\_DATE('2024-08-27', 'YYYY-MM-DD'), 1001, 'A\_103');

INSERT INTO BOOKINGS (BOOKING\_ID, BOOKING\_DATE, EVENT\_ID, ARTIST\_ID)

VALUES (4, TO\_DATE('2024-08-30', 'YYYY-MM-DD'), 1003, 'A\_101');

INSERT INTO BOOKINGS (BOOKING\_ID, BOOKING\_DATE, EVENT\_ID, ARTIST\_ID)

VALUES (5, TO\_DATE('2024-08-30', 'YYYY-MM-DD'), 1003, 'A\_102');

--QUESTION 1 : SQL QUERY FOR BOOKING DETAILS

SELECT

B.BOOKING\_ID,

E.EVENT\_NAME,

E.EVENT\_RATE,

A.ARTIST\_NAME,

A.ARTIST\_EMAIL

FROM

BOOKINGS B

JOIN EVENT E ON B.EVENT\_ID = E.EVENT\_ID

JOIN ARTIST A ON B.ARTIST\_ID = A.ARTIST\_ID;

--QUESTION 2: SQL QUERY FOR ARTIST WITH LEAST PERFORMANCES

SELECT

A.ARTIST\_ID,

A.ARTIST\_NAME,

A.ARTIST\_EMAIL,

COUNT(B.BOOKING\_ID) AS NUM\_PERFORMANCES

FROM

ARTIST A

LEFT JOIN BOOKINGS B ON A.ARTIST\_ID = B.ARTIST\_ID

GROUP BY

A.ARTIST\_ID,A.ARTIST\_NAME,A.ARTIST\_EMAIL

HAVING

COUNT(B.BOOKING\_ID) = (

SELECT MIN(COUNT(\*))

FROM BOOKINGS

GROUP BY ARTIST\_ID

);

--QUESTION 3: SQL QUERY FOR ARTIST REVENUE

SELECT

A.ARTIST\_NAME,

SUM(E.EVENT\_RATE) AS TOTAL\_REVENUE

FROM

BOOKINGS B

JOIN ARTIST A ON B.ARTIST\_ID = A.ARTIST\_ID

JOIN EVENT E ON B.EVENT\_ID = E.EVENT\_ID

GROUP BY

A.ARTIST\_NAME;

--QUESTION 4: PL/SQL QUERY FOR ARTIST AND BOOKING DATE

SET SERVEROUTPUT ON;

DECLARE

v\_artist\_name ARTIST.ARTIST\_NAME%TYPE;

v\_booking\_date BOOKINGS.BOOKING\_DATE%TYPE;

CURSOR booking\_cursor IS

SELECT A.ARTIST\_NAME, B.BOOKING\_DATE

FROM BOOKINGS B

JOIN ARTIST A ON B.ARTIST\_ID = A.ARTIST\_ID

WHERE B.EVENT\_ID = 1001;

BEGIN

FOR record IN booking\_cursor LOOP

v\_artist\_name := record.ARTIST\_NAME;

v\_booking\_date := record.BOOKING\_DATE;

DBMS\_OUTPUT.PUT\_LINE('ARTIST: ' || v\_artist\_name || ' | Booking Date: ' || TO\_CHAR(v\_booking\_date, 'YYYY-MM-DD'));

END LOOP;

END;

--QUESTION 5: PL/SQL QUERY FOR EVENT DISCOUNTS

SET SERVEROUTPUT ON;

DECLARE

v\_event\_name EVENT.EVENT\_NAME%TYPE;

v\_event\_rate EVENT.EVENT\_RATE%TYPE;

v\_discounted NUMBER;

CURSOR event\_cursor IS

SELECT EVENT\_NAME, EVENT\_RATE

FROM EVENT;

BEGIN

FOR rec IN event\_cursor LOOP

v\_event\_name := rec.EVENT\_NAME;

v\_event\_rate := rec.EVENT\_RATE;

IF v\_event\_rate > 250 THEN

v\_discounted := v\_event\_rate \* 0.9; -- 10% discount

DBMS\_OUTPUT.PUT\_LINE(

'Event: ' || v\_event\_name ||

' | Original Price: R' || v\_event\_rate ||

' | Discounted Price: R' || TO\_CHAR(ROUND(v\_discounted, 2))

);

ELSE

DBMS\_OUTPUT.PUT\_LINE(

'Event: ' || v\_event\_name ||

' | Price: R' || v\_event\_rate ||

' (No Discount)'

);

END IF;

END LOOP;

END;

--QUESTION 6 : CREATE EVENT\_SCHEDULES VIEW

CREATE OR REPLACE VIEW Event\_Schedules AS

SELECT

E.EVENT\_NAME,

B.BOOKING\_DATE

FROM

BOOKINGS B

JOIN EVENT E ON B.EVENT\_ID = E.EVENT\_ID

WHERE

B.BOOKING\_DATE BETWEEN TO\_DATE ('2024-07-01', 'YYYY-MM-DD')

AND TO\_DATE ('2024-08-28', 'YYYY-MM-DD');

SELECT \* FROM Event\_Schedules;

--QUESTION 7 : STORED PROCEDURE FOR BOOKING DETAILS

CREATE OR REPLACE PROCEDURE Get\_Bookings\_By\_Artist (

p\_artist\_name IN ARTIST.ARTIST\_NAME%TYPE

)

IS

BEGIN

FOR rec IN (

SELECT

B.BOOKING\_ID,

A.ARTIST\_NAME,

B.BOOKING\_DATE,

E.EVENT\_NAME,

E.EVENT\_RATE

FROM

BOOKINGS B

JOIN ARTIST A ON B.ARTIST\_ID = A.ARTIST\_ID

JOIN EVENT E ON B.EVENT\_ID = E.EVENT\_ID

WHERE

A.ARTIST\_NAME = p\_artist\_name

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Booking ID: ' || rec.BOOKING\_ID ||

' | Artist: ' || rec.ARTIST\_NAME ||

' | Date: ' || TO\_CHAR(rec.BOOKING\_DATE, 'YYYY-MM-DD') ||

' | Event: ' || rec.EVENT\_NAME ||

' | Rate: R' || rec.EVENT\_RATE);

END LOOP;

END;

SET SERVEROUTPUT ON;

BEGIN

Get\_Bookings\_By\_Artist('Lol Man');

END;

--QUESTION 8: CREATE A RELEVANT FUNCTION

--STEP 1

CREATE OR REPLACE FUNCTION Get\_Artist\_Revenue (

p\_artist\_name IN ARTIST.ARTIST\_NAME%TYPE

)

RETURN NUMBER

IS

v\_total\_revenue NUMBER := 0;

BEGIN

SELECT

NVL(SUM(E.EVENT\_RATE), 0)

INTO

v\_total\_revenue

FROM

BOOKINGS B

JOIN ARTIST A ON B.ARTIST\_ID = A.ARTIST\_ID

JOIN EVENT E ON B.EVENT\_ID = E.EVENT\_ID

WHERE

A.ARTIST\_NAME = p\_artist\_name;

RETURN v\_total\_revenue;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No bookings found for artist: ' || p\_artist\_name);

RETURN 0;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An unexpected error occurred: ' || SQLERRM);

RETURN -1;

END;

--STEP 2

SET SERVEROUTPUT ON;

DECLARE

v\_revenue NUMBER;

BEGIN

v\_revenue := Get\_Artist\_Revenue('Max Trillion');

IF v\_revenue = -1 THEN

DBMS\_OUTPUT.PUT\_LINE('Error occurred while calculating revenue.');

ELSIF v\_revenue = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Artist has no bookings.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Total revenue for Max Trillion: R' || v\_revenue);

END IF;

END;

--QUESTION 11: CREATE A TRIGGER TO PREVENT WEEKEND BOOKINGS

SELECT USER FROM DUAL;

CREATE USER C##oneuser IDENTIFIED BY strongpassword123;

GRANT CONNECT, RESOURCE TO C##oneuser;

CREATE OR REPLACE TRIGGER trg\_prevent\_invalid\_booking

BEFORE INSERT ON BOOKINGS

FOR EACH ROW

DECLARE

v\_day VARCHAR2(10);

BEGIN

v\_day := TO\_CHAR(:NEW.BOOKING\_DATE, 'DY', 'NLS\_DATE\_LANGUAGE=ENGLISH');

IF v\_day IN ('SAT', 'SUN') THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Bookings cannot be made on weekends.');

END IF;

END;

/

# Question 1 – Query for Booking Details

## SQL Code:

--QUESTION 1 : SQL QUERY FOR BOOKING DETAILS

SELECT

B.BOOKING\_ID,

E.EVENT\_NAME,

E.EVENT\_RATE,

A.ARTIST\_NAME,

A.ARTIST\_EMAIL

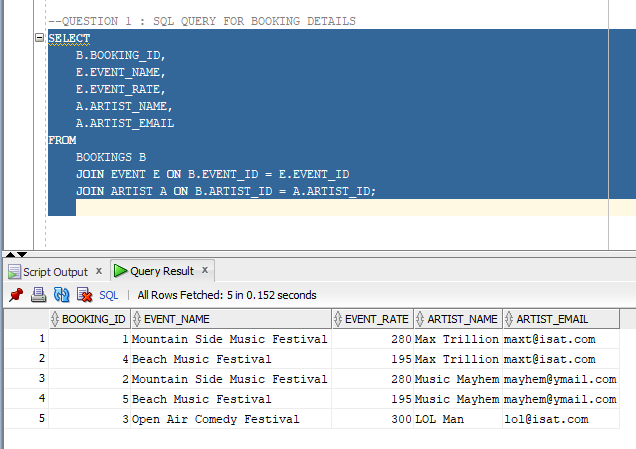
FROM

BOOKINGS B

JOIN EVENT E ON B.EVENT\_ID = E.EVENT\_ID

JOIN ARTIST A ON B.ARTIST\_ID = A.ARTIST\_ID;

## SQL Output:



# Question 2 – SQL Query for Artist with Least Performance

## SQL Code:

--QUESTION 2: SQL QUERY FOR ARTIST WITH LEAST PERFORMANCES

SELECT

A.ARTIST\_ID,

A.ARTIST\_NAME,

A.ARTIST\_EMAIL,

COUNT(B.BOOKING\_ID) AS NUM\_PERFORMANCES

FROM

ARTIST A

LEFT JOIN BOOKINGS B ON A.ARTIST\_ID = B.ARTIST\_ID

GROUP BY

A.ARTIST\_ID,A.ARTIST\_NAME,A.ARTIST\_EMAIL

HAVING

COUNT(B.BOOKING\_ID) = (

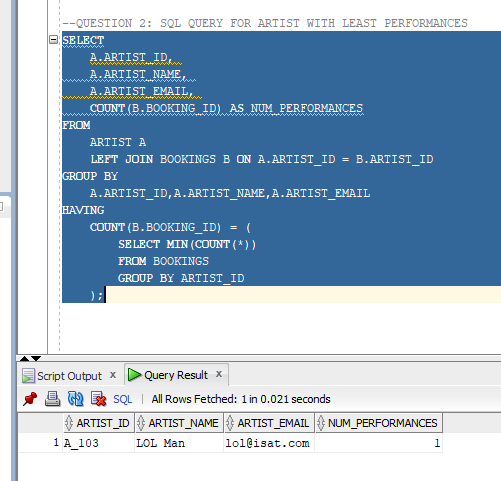
SELECT MIN(COUNT(\*))

FROM BOOKINGS

GROUP BY ARTIST\_ID

);

## SQL Output:



# Question 3 – SQL Query for Artist Revenue

## SQL Code:

--QUESTION 3: SQL QUERY FOR ARTIST REVENUE

SELECT

A.ARTIST\_NAME,

SUM(E.EVENT\_RATE) AS TOTAL\_REVENUE

FROM

BOOKINGS B

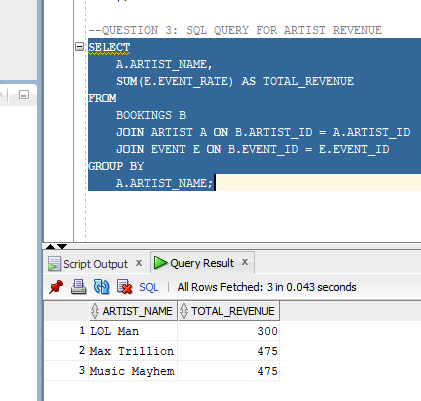
JOIN ARTIST A ON B.ARTIST\_ID = A.ARTIST\_ID

JOIN EVENT E ON B.EVENT\_ID = E.EVENT\_ID

GROUP BY

A.ARTIST\_NAME;

## SQL Output:



# Question 4- PL/SQL Query for Artist and Booking Date

## SQL Code:

--QUESTION 4: PL/SQL QUERY FOR ARTIST AND BOOKING DATE

SET SERVEROUTPUT ON;

DECLARE

v\_artist\_name ARTIST.ARTIST\_NAME%TYPE;

v\_booking\_date BOOKINGS.BOOKING\_DATE%TYPE;

CURSOR booking\_cursor IS

SELECT A.ARTIST\_NAME, B.BOOKING\_DATE

FROM BOOKINGS B

JOIN ARTIST A ON B.ARTIST\_ID = A.ARTIST\_ID

WHERE B.EVENT\_ID = 1001;

BEGIN

FOR record IN booking\_cursor LOOP

v\_artist\_name := record.ARTIST\_NAME;

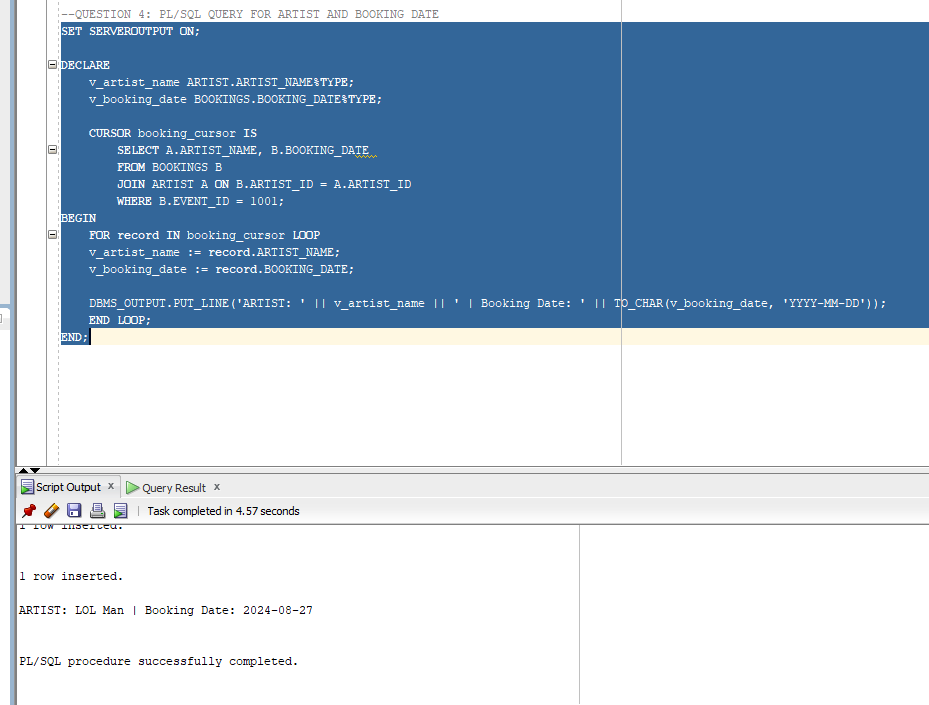
v\_booking\_date := record.BOOKING\_DATE;

DBMS\_OUTPUT.PUT\_LINE('ARTIST: ' || v\_artist\_name || ' | Booking Date: ' || TO\_CHAR(v\_booking\_date, 'YYYY-MM-DD'));

END LOOP;

END;

## SQL Output:



# Question 5 – PL/SQL Query for Event Discounts

## SQL Code:

--QUESTION 5: PL/SQL QUERY FOR EVENT DISCOUNTS

SET SERVEROUTPUT ON;

DECLARE

v\_event\_name EVENT.EVENT\_NAME%TYPE;

v\_event\_rate EVENT.EVENT\_RATE%TYPE;

v\_discounted NUMBER;

CURSOR event\_cursor IS

SELECT EVENT\_NAME, EVENT\_RATE

FROM EVENT;

BEGIN

FOR rec IN event\_cursor LOOP

v\_event\_name := rec.EVENT\_NAME;

v\_event\_rate := rec.EVENT\_RATE;

IF v\_event\_rate > 250 THEN

v\_discounted := v\_event\_rate \* 0.9; -- 10% discount

DBMS\_OUTPUT.PUT\_LINE(

'Event: ' || v\_event\_name ||

' | Original Price: R' || v\_event\_rate ||

' | Discounted Price: R' || TO\_CHAR(ROUND(v\_discounted, 2))

);

ELSE

DBMS\_OUTPUT.PUT\_LINE(

'Event: ' || v\_event\_name ||

' | Price: R' || v\_event\_rate ||

' (No Discount)'

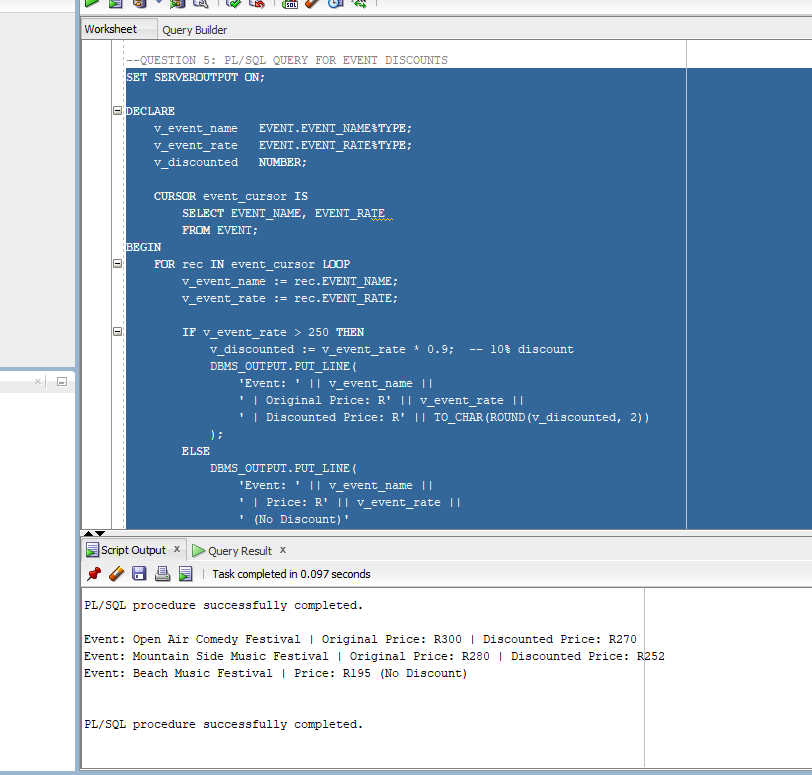
);

END IF;

END LOOP;

END; /

## SQL Output:



# Question 6 – Create Event\_Schedules View

## SQL Code:

--QUESTION 6 : CREATE EVENT\_SCHEDULES VIEW

CREATE OR REPLACE VIEW Event\_Schedules AS

SELECT

E.EVENT\_NAME,

B.BOOKING\_DATE

FROM

BOOKINGS B

JOIN EVENT E ON B.EVENT\_ID = E.EVENT\_ID

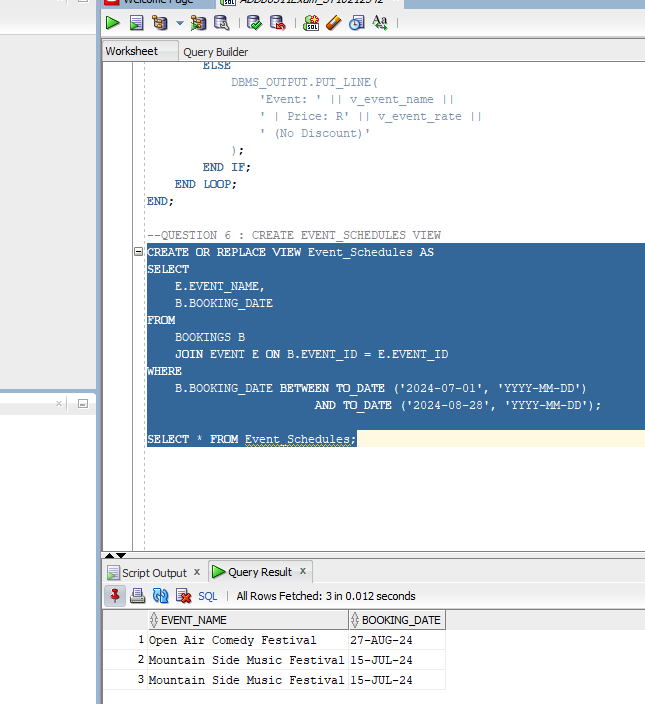
WHERE

B.BOOKING\_DATE BETWEEN TO\_DATE ('2024-07-01', 'YYYY-MM-DD')

AND TO\_DATE ('2024-08-28', 'YYYY-MM-DD');

SELECT \* FROM Event\_Schedules;

## SQL Output:



# Question 7 – Stored Procedure for Booking Details

## SQL Code:

--QUESTION 7 : STORED PROCEDURE FOR BOOKING DETAILS

CREATE OR REPLACE PROCEDURE Get\_Bookings\_By\_Artist (

p\_artist\_name IN ARTIST.ARTIST\_NAME%TYPE

)

IS

BEGIN

FOR rec IN (

SELECT

B.BOOKING\_ID,

A.ARTIST\_NAME,

B.BOOKING\_DATE,

E.EVENT\_NAME,

E.EVENT\_RATE

FROM

BOOKINGS B

JOIN ARTIST A ON B.ARTIST\_ID = A.ARTIST\_ID

JOIN EVENT E ON B.EVENT\_ID = E.EVENT\_ID

WHERE

A.ARTIST\_NAME = p\_artist\_name

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Booking ID: ' || rec.BOOKING\_ID ||

' | Artist: ' || rec.ARTIST\_NAME ||

' | Date: ' || TO\_CHAR(rec.BOOKING\_DATE, 'YYYY-MM-DD') ||

' | Event: ' || rec.EVENT\_NAME ||

' | Rate: R' || rec.EVENT\_RATE);

END LOOP;

END;

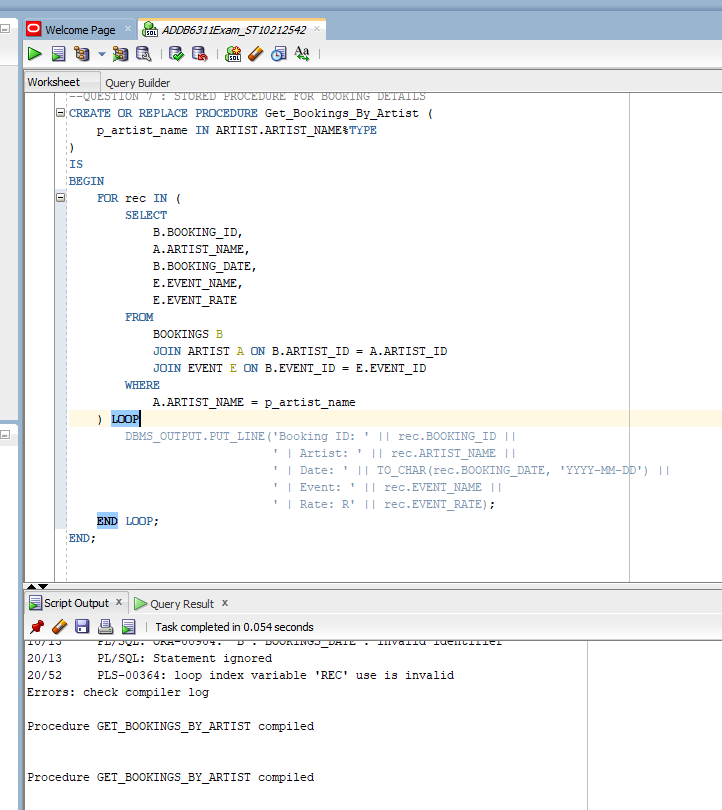
SET SERVEROUTPUT ON;

BEGIN

Get\_Bookings\_By\_Artist('Lol Man');

END;

## SQL Output

A screenshot of a computer

AI-generated content may be incorrect.

# Question 8 – Create a relevant function

## SQL Code:

--QUESTION 8: CREATE A RELEVANT FUNCTION

--STEP 1

CREATE OR REPLACE FUNCTION Get\_Artist\_Revenue (

p\_artist\_name IN ARTIST.ARTIST\_NAME%TYPE

)

RETURN NUMBER

IS

v\_total\_revenue NUMBER := 0;

BEGIN

SELECT

NVL(SUM(E.EVENT\_RATE), 0)

INTO

v\_total\_revenue

FROM

BOOKINGS B

JOIN ARTIST A ON B.ARTIST\_ID = A.ARTIST\_ID

JOIN EVENT E ON B.EVENT\_ID = E.EVENT\_ID

WHERE

A.ARTIST\_NAME = p\_artist\_name;

RETURN v\_total\_revenue;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No bookings found for artist: ' || p\_artist\_name);

RETURN 0;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An unexpected error occurred: ' || SQLERRM);

RETURN -1;

END;

--STEP 2

SET SERVEROUTPUT ON;

DECLARE

v\_revenue NUMBER;

BEGIN

v\_revenue := Get\_Artist\_Revenue('Max Trillion');

IF v\_revenue = -1 THEN

DBMS\_OUTPUT.PUT\_LINE('Error occurred while calculating revenue.');

ELSIF v\_revenue = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Artist has no bookings.');

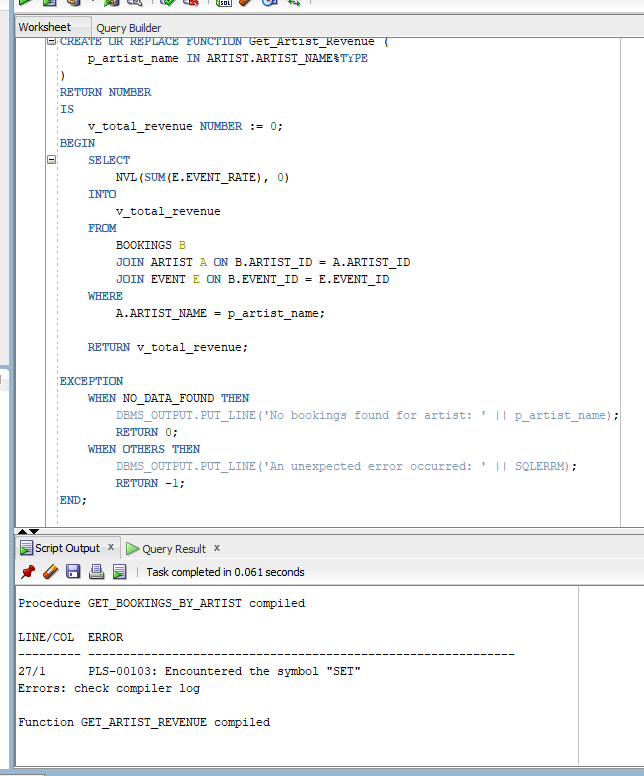
ELSE

DBMS\_OUTPUT.PUT\_LINE('Total revenue for Max Trillion: R' || v\_revenue);

END IF;

END;

## SQL Output:

A screenshot of a computer

AI-generated content may be incorrect.

# Question 9 – Database Security Tools and Platforms

Tools and Platforms that were used to excute the creation of this database include the following:

* **Data Encryption**

Since the database handles personal details like ARTIST\_EMAIL, data encryption is critical. Oracle Advanced Security helps you encrypt sensitive data both at rest and in transit, ensuring protection even if unauthorized access occurs (Oracle, 2023a).

* **Oracle Data Safe**

This is particularly useful for assessing how securely users interact with the database. Given your schema includes relationships between events and artists, you can audit user behavior, detect anomalies, and mask sensitive data when accessed for testing or development (Oracle, 2023b).

* **Oracle Audit Vault and Database Firewall**

To prevent SQL injection or any malicious activity—especially since stored procedures like Get\_Bookings\_By\_Artist run dynamic queries—this tool logs and monitors all activity, raising flags for suspicious operations (Oracle, 2023c).

* **Access Control**

Only authorized roles should be allowed to modify booking information or access artist contact data. IAM ensures role-based access control, single sign-on, and strong authentication policies (Oracle, 2023d).

* **Oracle Cloud Infrastructure (OCI) Security**

If you deploy the database to the cloud, OCI security features provide network isolation, compartmentalization, and encryption key management. These are essential for compliance and resilience against external threats (Oracle, 2023e).

# Question 10 – PL/SQL Scalar Data Types Discussion

In PL/SQL, scalar data types are fundamental building blocks used to store single, indivisible values. Examples of these include the following:

1. Numeric Types: Used to store numbers, both integers and floating-point values. In the case of the practicum , event rates.

For example:

v\_rate EVENT.EVENT\_RATE%TYPE; -- Decimal(10,2)

You can also define with native types: v\_discounted NUMBER := 250.50;

1. Character Types: Store alphanumeric data such as names, descriptions, and email addresses. Common types to identify them include:
   1. CHAR
   2. VARCHAR2
   3. CLOB

For example:

v\_artist\_name ARTIST.ARTIST\_NAME%TYPE; -- VARCHAR2

v\_email VARCHAR2(100) := 'test@example.com';

1. Date/Time Types: Store temporal data such as booking and event dates. The DATE type includes both date and time components. PL/SQL supports date artithmetic and formatting.

For example;

IF TO\_CHAR(v\_booking\_date, 'DY') = 'SAT' THEN

DBMS\_OUTPUT.PUT\_LINE('Weekend booking');

END IF;

1. Boolean Type: Stores logical values: TRUE,FALSE, or NULL. It is used for control flow and validation logic:

For example:

v\_is\_valid BOOLEAN := TRUE;

These types help manage data with appropriate validation and formatting in the case study—e.g., DATE for BOOKING\_DATE, NUMBER for EVENT\_RATE, and VARCHAR2 for ARTIST\_EMAIL.

# Question 11 – Create a Trigger to Prevent Weekend Bookings

## SQL Code:

--QUESTION 11: CREATE A TRIGGER TO PREVENT WEEKEND BOOKINGS

SELECT USER FROM DUAL;

CREATE USER C##oneuser IDENTIFIED BY strongpassword123;

GRANT CONNECT, RESOURCE TO C##oneuser;

CREATE OR REPLACE TRIGGER trg\_prevent\_invalid\_booking

BEFORE INSERT ON BOOKINGS

FOR EACH ROW

DECLARE

v\_day VARCHAR2(10);

BEGIN

v\_day := TO\_CHAR(:NEW.BOOKING\_DATE, 'DY', 'NLS\_DATE\_LANGUAGE=ENGLISH');

IF v\_day IN ('SAT', 'SUN') THEN

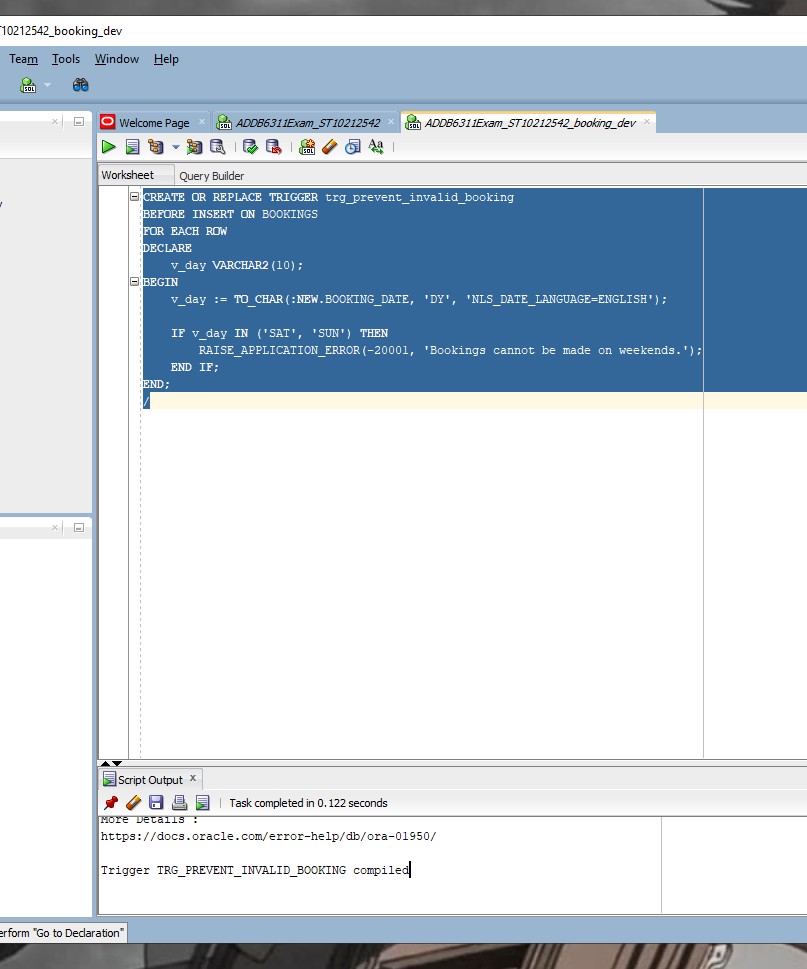
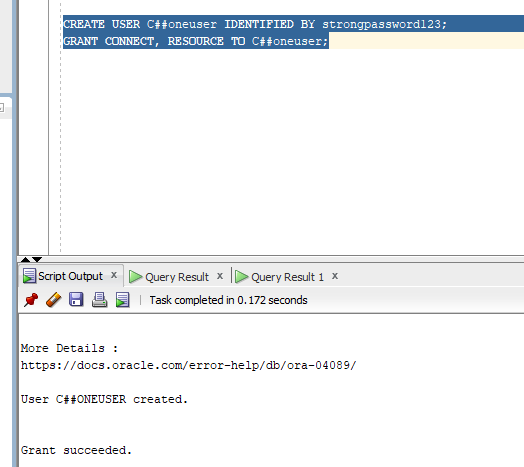
RAISE\_APPLICATION\_ERROR(-20001, 'Bookings cannot be made on weekends.');

END IF;

END;

/

## SQL Output:



# GitHub Link:

<https://github.com/AaliyahAllie/ADDB63111_PRACTICUM_ST10212542.git>

# References:

Oracle. (2023a) Oracle Advanced Security. Available at: https://www.oracle.com/security/database-security/advanced-security/ (Accessed: 18 June 2025).

Oracle. (2023b) Oracle Data Safe. Available at: https://www.oracle.com/security/database-security/data-safe/ (Accessed: 18 June 2025).

Oracle. (2023c) Oracle Audit Vault and Database Firewall. Available at: https://www.oracle.com/security/database-security/audit-vault-database-firewall/ (Accessed: 18 June 2025).

Oracle. (2023d) Oracle Identity Management. Available at: https://www.oracle.com/security/identity-management/ (Accessed: 18 June 2025).

Oracle. (2023e) Oracle Cloud Infrastructure Security. Available at: https://www.oracle.com/cloud/security/ (Accessed: 18 June 2025).

Oracle. (2023a) PL/SQL Data Types. Available at: https://docs.oracle.com/en/database/oracle/oracle-database/23/lnpls/plsql-data-types.html (Accessed: 18 June 2025).

ComplexSQL. (2023) PL SQL Scalar Datatypes. Available at: https://www.complexsql.com/scalar-datatypes/ (Accessed: 18 June 2025).

TutorialsPoint. (2023) PL/SQL Data Types and Subtypes. Available at: https://www.tutorialspoint.com/plsql/plsql\_data\_types.htm (Accessed: 18 June 2025).