American International University-Bangladesh

Final Term Project Documentation

Project Name: welfare club Management System

Course Name: Advance Database Management System

Section: B

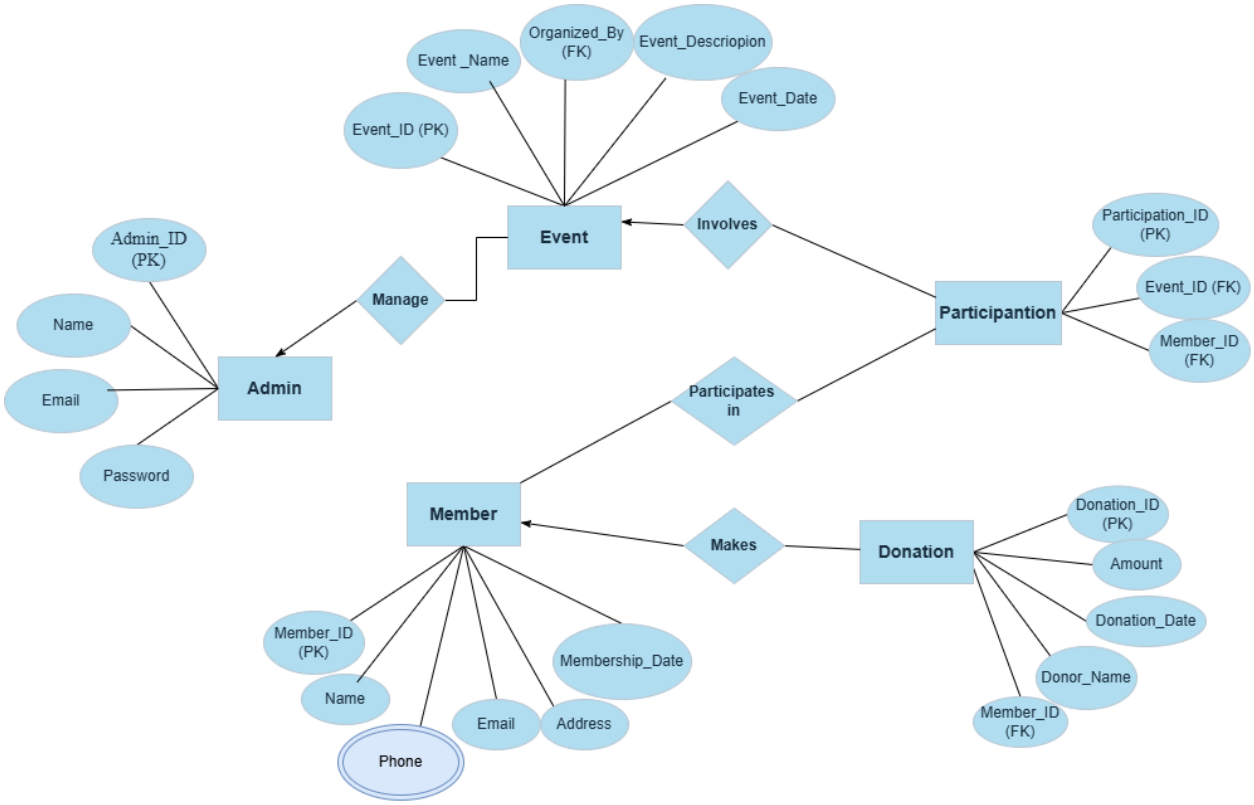
Serial No	Name	ID	Contributed Percentage	Topic Names
01	MD. ABDULLAH AL MAHMUD NAIEM	21-45255-2	25%	Project planning, project proposal, Relational Algebra, Report writing
02	SUMAIYA TAB	21-45928-3	25%	Pl/SQL mid, Final
03	MST.JANNATUL FERDOUS	21-45440-3	25%	Class Diagram, Use Case Diagram, Activity Diagram, Schema Diagram
04	MD. ABDULLAHEL RAFI	19-41051-2	25%	User Interface design

Project Updates

Scenario Description:

The Welfare Club Management System simplifies and organizes welfare club operations through a centralized platform for administrators and members. When a member joins the Welfare Club, their details are stored in the system, including Member_ID, Name, Phone, Email, Address, and membership_date. The admin has the authority to add new members and manage existing ones, ensuring the club maintains an active and engaged community. The member table records all registered members and links them to their participation in events and donations. There is also a admin table for Admins that contains Admin_ID, Name, Email, Password. Events are a core part of the Welfare Club, and the system allows Admins to create and manage them efficiently. Each event is assigned a unique Event_ID and includes details such as the Event_Name, Event_Date, and description. Members can see upcoming events and register their participation through the system. The event Participation table records Participation_ID, Event_ ID and Member_ID ensuring an accurate log of participation.Financial contributions play a significant role in supporting club activities. Members can donate funds, which are recorded in the donation table. This table includes donation_ID, Member_ID, Donor_Name, Donation_Date and the donated amount. Admin can track and manage these financial transactions, ensuring transparency and accountability in fund utilization.The Welfare Club Management System ensures a streamlined approach to managing club operations, reducing manual workload and minimizing errors. Members can actively engage with the club by registering for events and making contributions, while Admin can efficiently organize activities and maintain financial transparency. By integrating technology into welfare club management, the system enhances operational efficiency and fosters a well-organized, engaged community.

ER diagram:



Normalization:

Manage

UNF:

Event_ID (PK), Event_Name, Event_Description, Event_Date, Organized_By (FK), Admin_ID (PK), Name, Email, Password

1NF: No multi-valued attribute.

Event_ID (PK), Event_Name, Event_Description, Event_Date, Organized_By (FK), Admin_ID (PK), Name, Email, Password

2NF:

1. Event_ID (PK), Event_Name, Event_Description, Event_Date, Organized_By (FK)
2. Admin_ID (PK), Name, Email, Password

3NF: There is no transitive dependency. Relation already in 3NF.

1. Event_ID (PK), Event_Name, Event_Description, Event_Date, Organized_By (FK)
2. Admin_ID (PK), Name, Email, Password

Table Creation:

1. Event_ID (PK), Event_Name, Event_Description, Event_Date, Organized_By (FK)
2. Admin_ID (PK), Name, Email, Password

Involves

UNF:

Event_ID (PK), Event_Name, Event_Description, Event_Date,
Organized_By(FK),Participation_ID (PK), Event_ID (FK), Member_ID (FK)

1NF: No multi-valued attribute

Event_ID (PK), Event_Name, Event_Description, Event_Date,
Organized_By(FK),Participation_ID (PK), Event_ID (FK), Member_ID (FK)

2NF:

1. Event_ID (PK), Event_Name, Event_Description, Event_Date, Organized_By(FK)
2. Participation_ID (PK), Event_ID (FK), Member_ID (FK)

3NF: There is no transitive dependency. Relation already in 3NF.

1. Event_ID (PK), Event_Name, Event_Description, Event_Date, Organized_By(FK)
2. Participation_ID (PK), Event_ID (FK), Member_ID (FK)

Table Creation:

1. Event_ID (PK), Event_Name, Event_Description, Event_Date, Organized_By(FK)
2. Participation_ID (PK), Event_ID (FK), Member_ID (FK)

Participates In

UNF:

Participation_ID (PK), Event_ID (FK), Member_ID (FK), Member_ID (PK), Name, Email, Address, Phone, Membership_Date

1NF:

Phone is a multi-valued attribute

Participation_ID (PK), Event_ID (FK), Member_ID (FK), Member_ID (PK), Name, Email, Address, Phone, Membership_Date

2NF:

1. Participation_ID (PK), Event_ID (FK), Member_ID (FK),
2. Member_ID (PK), Name, Email, Address, Phone, Membership_Date

3NF: There is no transitive dependency. Relation already in 3NF.

1. Participation_ID (PK), Event_ID (FK), Member_ID (FK),
2. Member_ID (PK), Name, Email, Address, Phone, Membership_Date

Table Creation:

1. Participation_ID (PK), Event_ID (FK), Member_ID (FK),
2. Member_ID (PK), Name, Email, Address, Phone, Membership_Date

Makes:

UNF:

Member_ID (PK), Name, Email, Address, Phone, Membership_Date, Donation_ID (PK), Amount, Donation_Date, Donor_Name, Member_ID (FK)

1NF:

Phone is a multi-valued attribute

Member_ID (PK), Name, Email, Address, Phone, Membership_Date, Donation_ID (PK), Amount, Donation_Date, Donor_Name, Member_ID (FK)

2NF:

1. Member_ID (PK), Name, Email, Address, Phone, Membership_Date
2. Donation_ID (PK), Amount, Donation_Date, Donor_Name, Member_ID (FK)

3. **3NF:** There is no transitive dependency. Relation already in 3NF.
1. Member_ID (PK), Name, Email, Address, Phone, Membership_Date
2. Donation_ID (PK), Amount, Donation_Date, Donor_Name, Member_ID (FK)

Table Creation:

1. Member_ID (PK), Name, Email, Address, Phone, Membership_Date
2. Donation_ID (PK), Amount, Donation_Date, Donor_Name, Member_ID (FK)

Temporary Tables

1. Event_ID (PK), Event_Name, Event_Description, Event_Date, Organized_By (FK)
2. Admin_ID (PK), Name, Email, Password
3. ~~Event_ID (PK), Event_Name, Event_Description, Event_Date, Organized_By(FK)~~
4. Participation_ID (PK), Event_ID (FK), Member_ID (FK)
5. ~~Participation_ID (PK), Event_ID (FK), Member_ID (FK),~~
6. Member_ID (PK), Name, Email, Address, Phone, Membership_Date
7. ~~Member_ID (PK), Name, Email, Address, Phone, Membership_Date~~
8. Donation_ID (PK), Amount, Donation_Date, Donor_Name, Member_ID (FK)

Final Tables

1. Event_ID (PK), Event_Name, Event_Description, Event_Date, Organized_By (FK)
2. Admin_ID (PK), Name, Email, Password
3. Participation_ID (PK), Event_ID (FK), Member_ID (FK)
4. Member_ID (PK), Name, Email, Address, Phone, Membership_Date
5. Donation_ID (PK), Amount, Donation_Date, Donor_Name, Member_ID (FK)

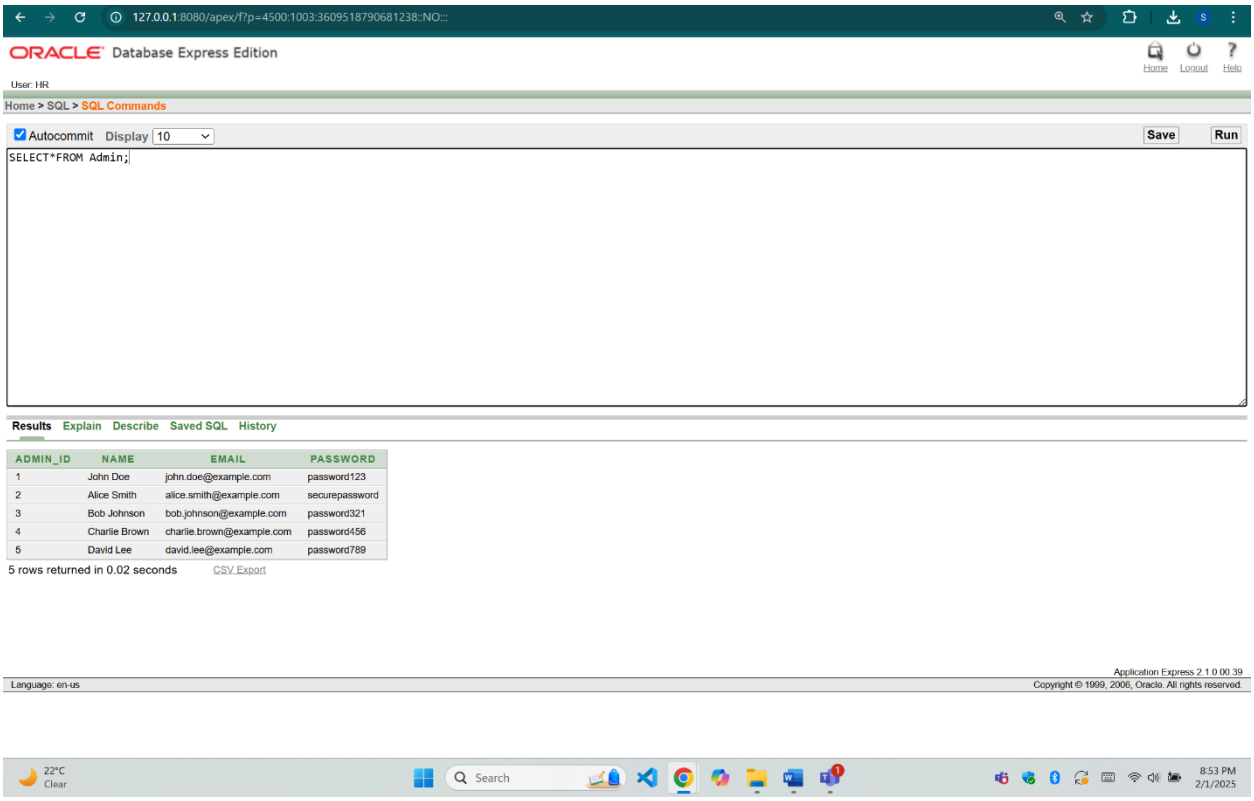
Table Creation:

```
CREATE TABLE Admin (
Admin_ID INT PRIMARY KEY,
   Name VARCHAR(255) NOT NULL,
   Email VARCHAR(255) UNIQUE NOT NULL,
   Password VARCHAR(255) NOT NULL
);

CREATE TABLE Event (
Event_ID INT PRIMARY KEY,
```

```
Event_Name VARCHAR2(255) NOT NULL,
Event_Description CLOB,  -- Replacing TEXT with CLOB
Event_Date DATE NOT NULL,
Organized_By INT,
   FOREIGN KEY (Organized_By) REFERENCES Admin(Admin_ID)
);

CREATE TABLE Member (
Member_ID INT PRIMARY KEY,
   Name VARCHAR2(255) NOT NULL,
   Email VARCHAR2(255) UNIQUE NOT NULL,
   Address CLOB,  -- Replacing TEXT with CLOB
   Phone VARCHAR2(20),
Membership_Date DATE NOT NULL
);

CREATE TABLE Participation (
Participation_ID NUMBER PRIMARY KEY,
Event_ID NUMBER,
Member_ID NUMBER,
   FOREIGN KEY (Event_ID) REFERENCES Event(Event_ID),
   FOREIGN KEY (Member_ID) REFERENCES Member(Member_ID)
);

CREATE TABLE Donation (
Donation_ID INT PRIMARY KEY,
   Amount DECIMAL(10,2) NOT NULL,
Donation_Date DATE NOT NULL,
Donor_Name VARCHAR(255) NOT NULL,
Member_ID INT,
   FOREIGN KEY (Member_ID) REFERENCES Member(Member_ID)
);
```

Data Insertion:

```
INSERT INTO Admin (Admin_ID, Name, Email, Password) VALUES

(1, 'John Doe', 'john.doe@example.com', 'password123'),

(2, 'Alice Smith', 'alice.smith@example.com', 'securepassword'),

(3, 'Bob Johnson', 'bob.johnson@example.com', 'password321'),

(4, 'Charlie Brown', 'charlie.brown@example.com', 'password456'),

(5, 'David Lee', 'david.lee@example.com', 'password789');
```



INSERT INTO Event (Event_ID, Event_Name, Event_Description, Event_Date, Organized_By) VALUES

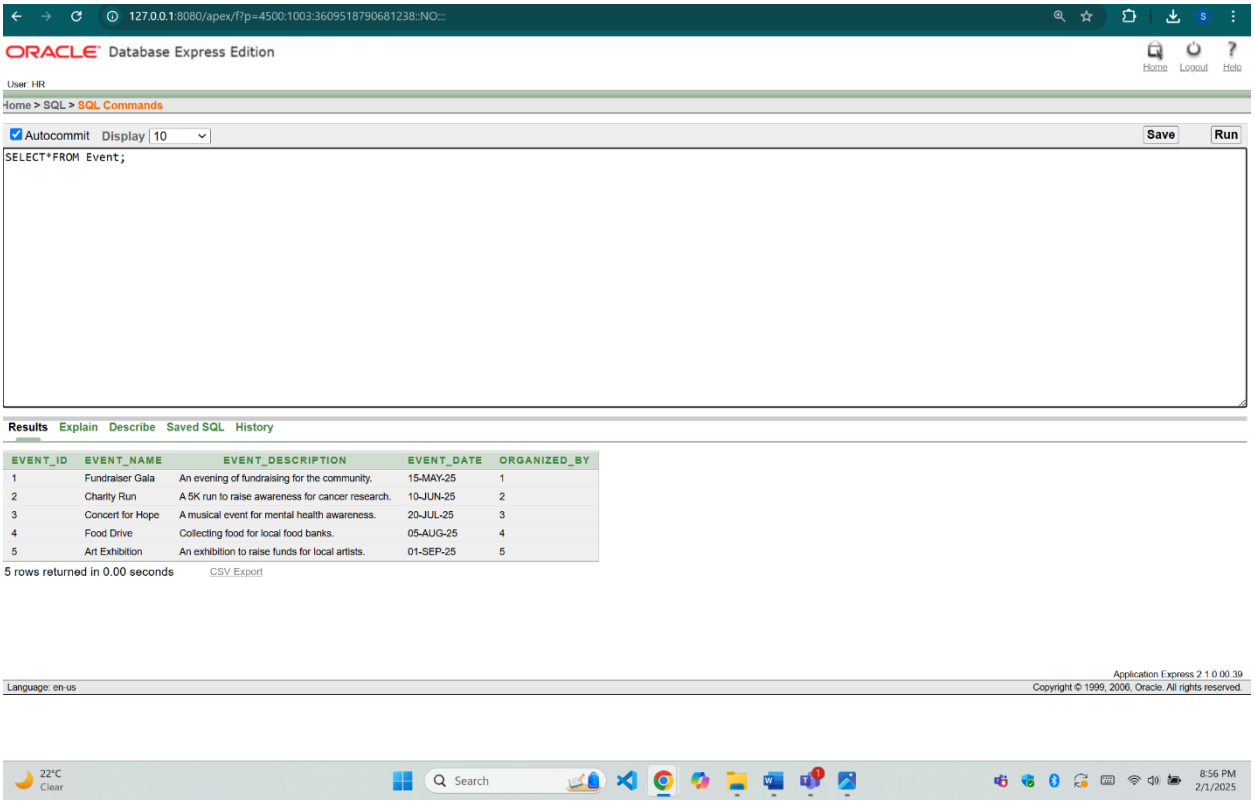
(1, 'Fundraiser Gala', 'An evening of fundraising for the community.', TO_DATE('2025-05-15', 'YYYY-MM-DD'), 1),

(2, 'Charity Run', 'A 5K run to raise awareness for cancer research.', TO_DATE('2025-06-10', 'YYYY-MM-DD'), 2),

(3, 'Concert for Hope', 'A musical event for mental health awareness.', TO_DATE('2025-07-20', 'YYYY-MM-DD'), 3),

(4, 'Food Drive', 'Collecting food for local food banks.', TO_DATE('2025-08-05', 'YYYY-MM-DD'), 4),

(5, 'Art Exhibition', 'An exhibition to raise funds for local artists.', TO_DATE('2025-09-01', 'YYYY-MM-DD'), 5);



INSERT INTO Member (Member_ID, Name, Email, Address, Phone, Membership_Date) VALUES

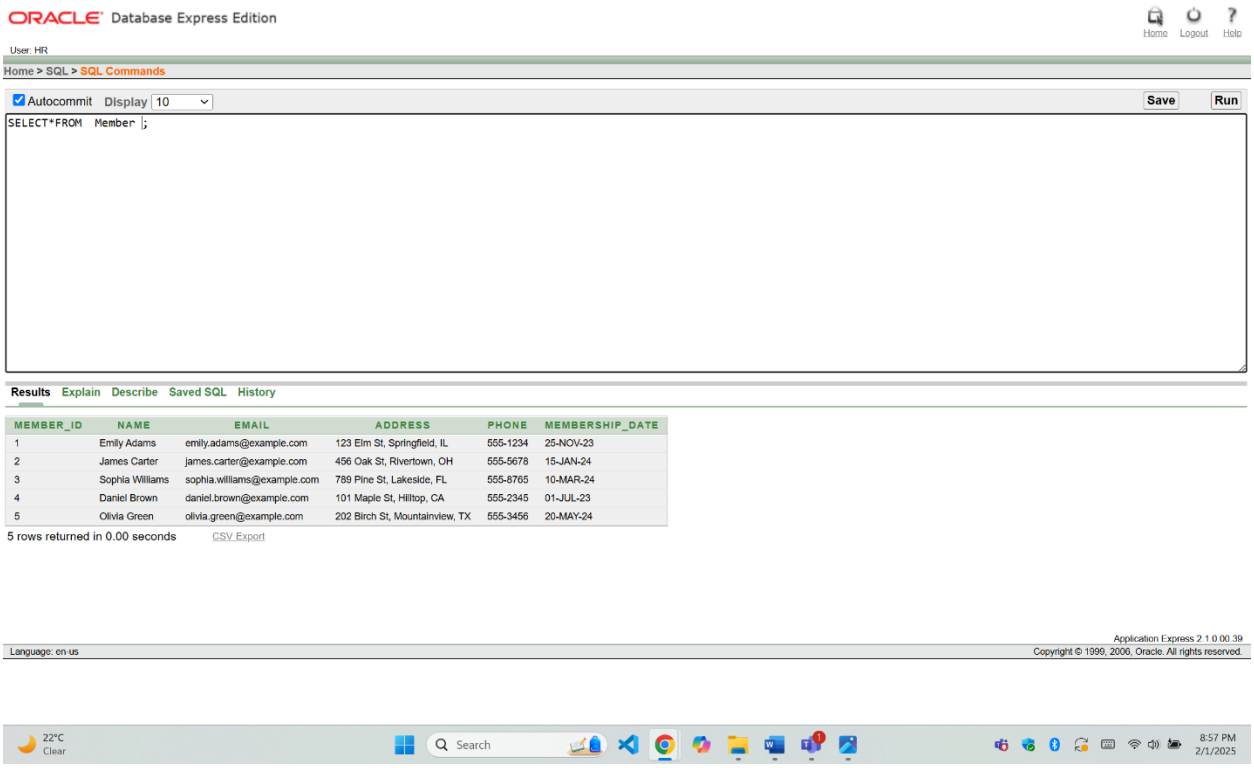
(1, 'Emily Adams', 'emily.adams@example.com', '123 Elm St, Springfield, IL', '555-1234', TO_DATE('2023-11-25', 'YYYY-MM-DD')),

(2, 'James Carter', 'james.carter@example.com', '456 Oak St, Rivertown, OH', '555-5678', TO_DATE('2024-01-15', 'YYYY-MM-DD')),

(3, 'Sophia Williams', 'sophia.williams@example.com', '789 Pine St, Lakeside, FL', '555-8765', TO_DATE('2024-03-10', 'YYYY-MM-DD')),

(4, 'Daniel Brown', 'daniel.brown@example.com', '101 Maple St, Hilltop, CA', '555-2345', TO_DATE('2023-07-01', 'YYYY-MM-DD')),

(5, 'Olivia Green', 'olivia.green@example.com', '202 Birch St, Mountainview, TX', '555-3456', TO_DATE('2024-05-20', 'YYYY-MM-DD'));



INSERT INTO Participation (Participation_ID, Event_ID, Member_ID) VALUES

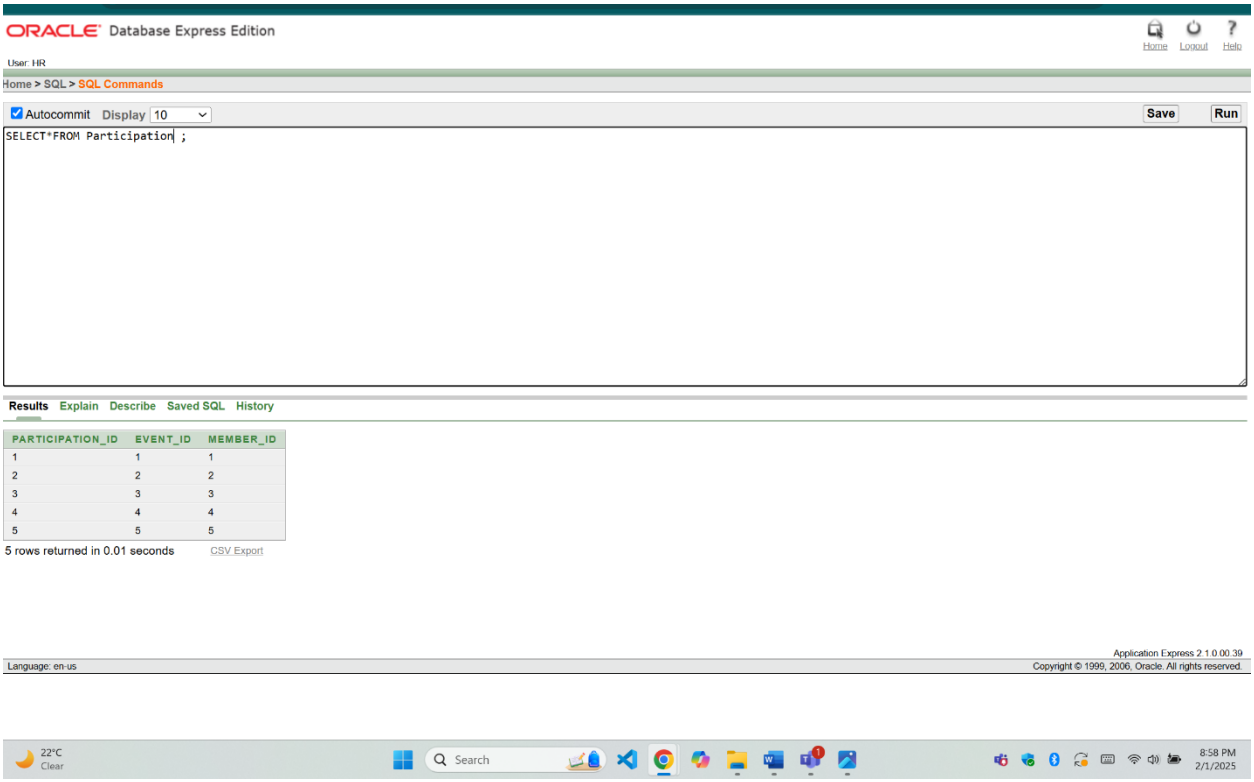
(1, 1, 1),

(2, 2, 2),

(3, 3, 3),

(4, 4, 4),

(5, 5, 5);



INSERT INTO Donation (Donation_ID, Amount, Donation_Date, Donor_Name, Member_ID) VALUES

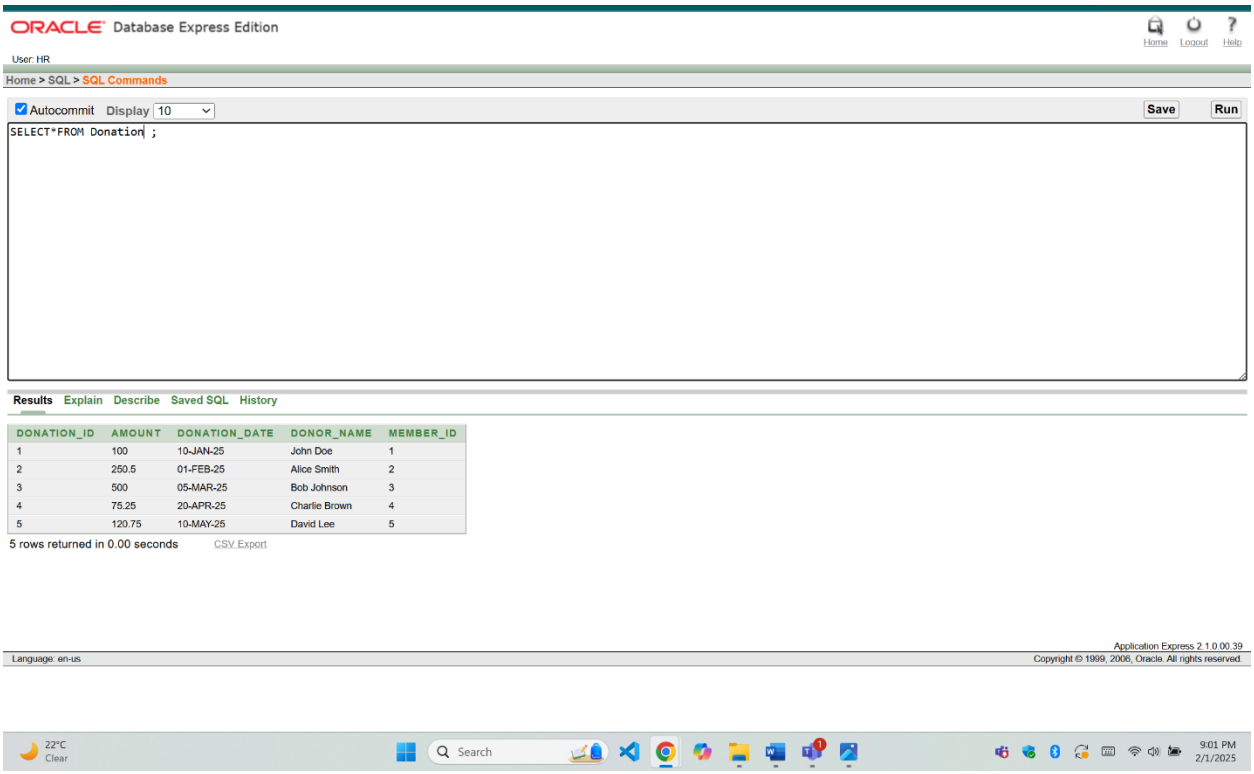
(1, 100.00, TO_DATE('2025-01-10', 'YYYY-MM-DD'), 'John Doe', 1),

(2, 250.50, TO_DATE('2025-02-01', 'YYYY-MM-DD'), 'Alice Smith', 2),

(3, 500.00, TO_DATE('2025-03-05', 'YYYY-MM-DD'), 'Bob Johnson', 3),

(4, 75.25, TO_DATE('2025-04-20', 'YYYY-MM-DD'), 'Charlie Brown', 4),

(5, 120.75, TO_DATE('2025-05-10', 'YYYY-MM-DD'), 'David Lee', 5);



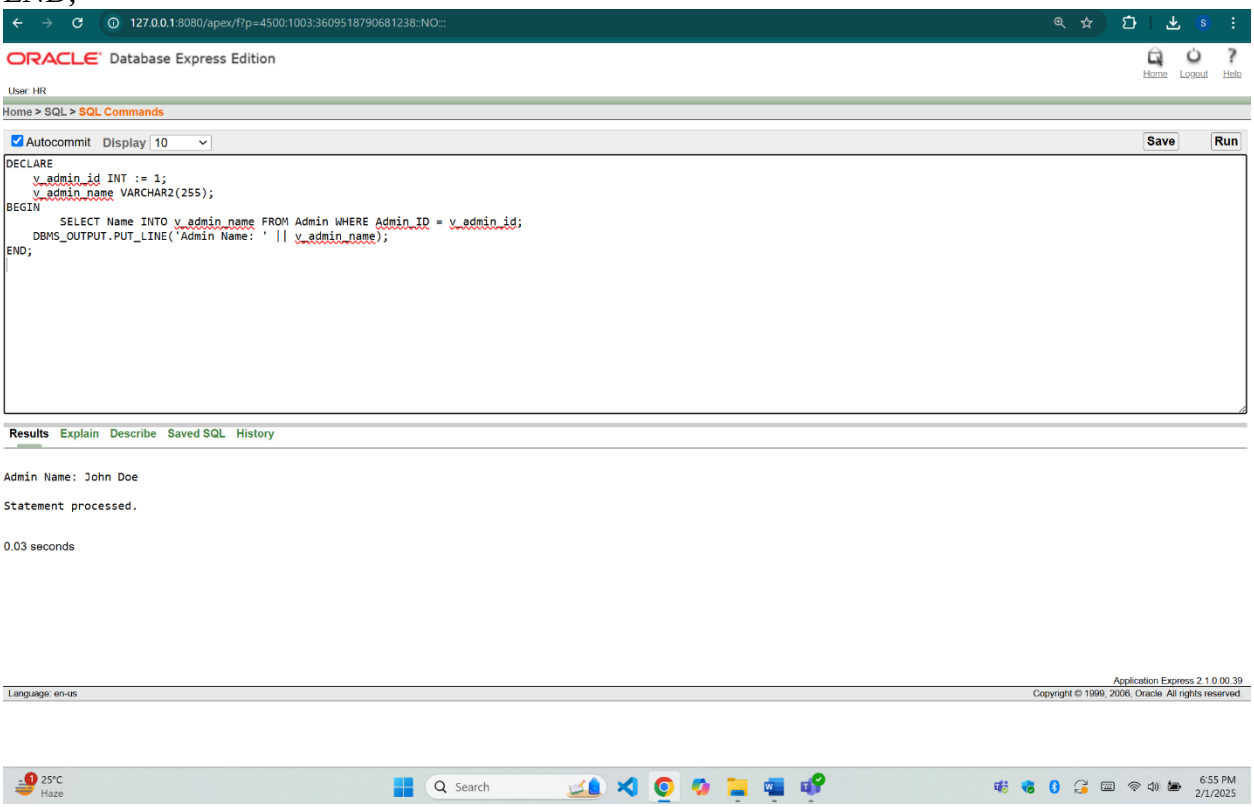
1. Variables (PL/SQL)

Query 1:

Question: How does the PL/SQL block retrieve and display the name of the admin using a variable `v_admin_name`?

```
DECLARE

v_admin_id INT := 1;

v_admin_name VARCHAR2(255);

BEGIN

    SELECT Name INTO v_admin_name FROM Admin WHERE Admin_ID = v_admin_id;

  DBMS_OUTPUT.PUT_LINE('Admin Name: ' || v_admin_name);
```

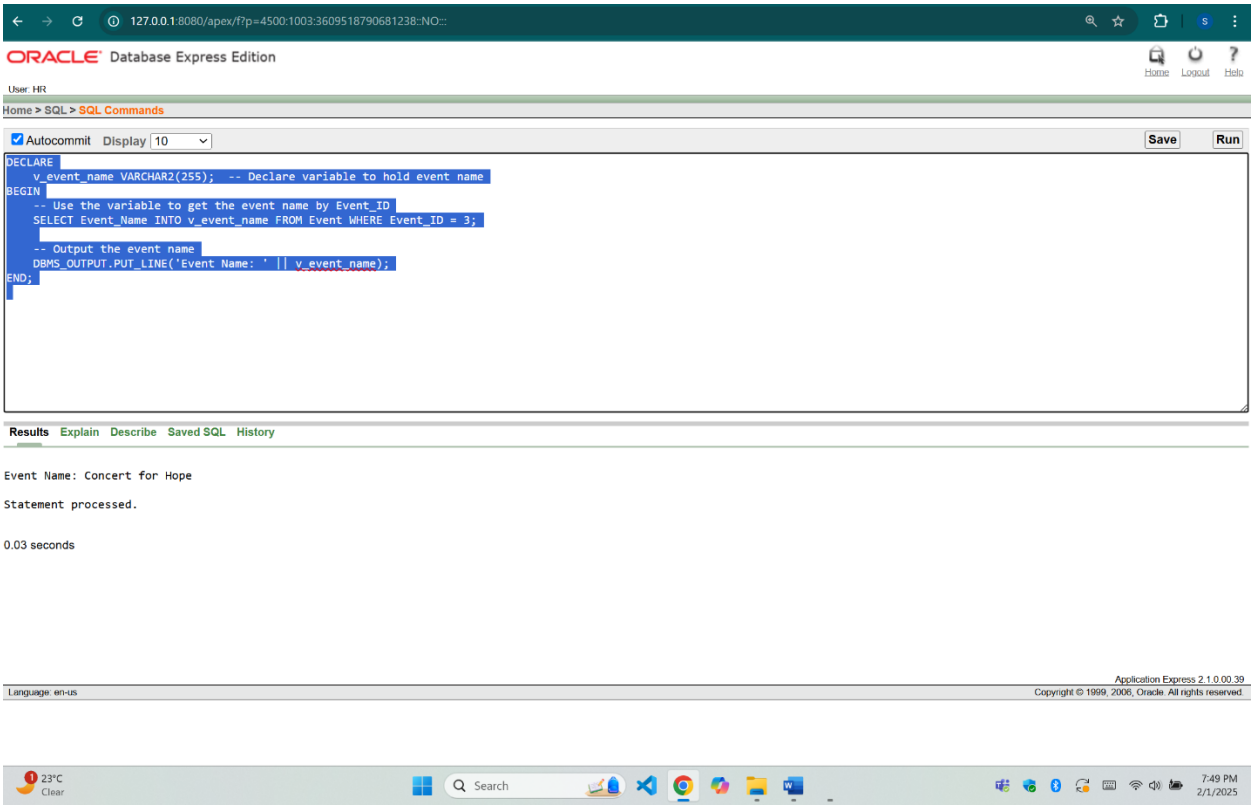
END;



Query 2:

Question: How does the PL/SQL block fetch and display the name of the event with Event_ID 3?

```
DECLARE
v_event_name VARCHAR2(255);
BEGIN

    SELECT Event_Name INTO v_event_name FROM Event WHERE Event_ID = 3;


    DBMS_OUTPUT.PUT_LINE('Event Name: ' || v_event_name);
END;
```

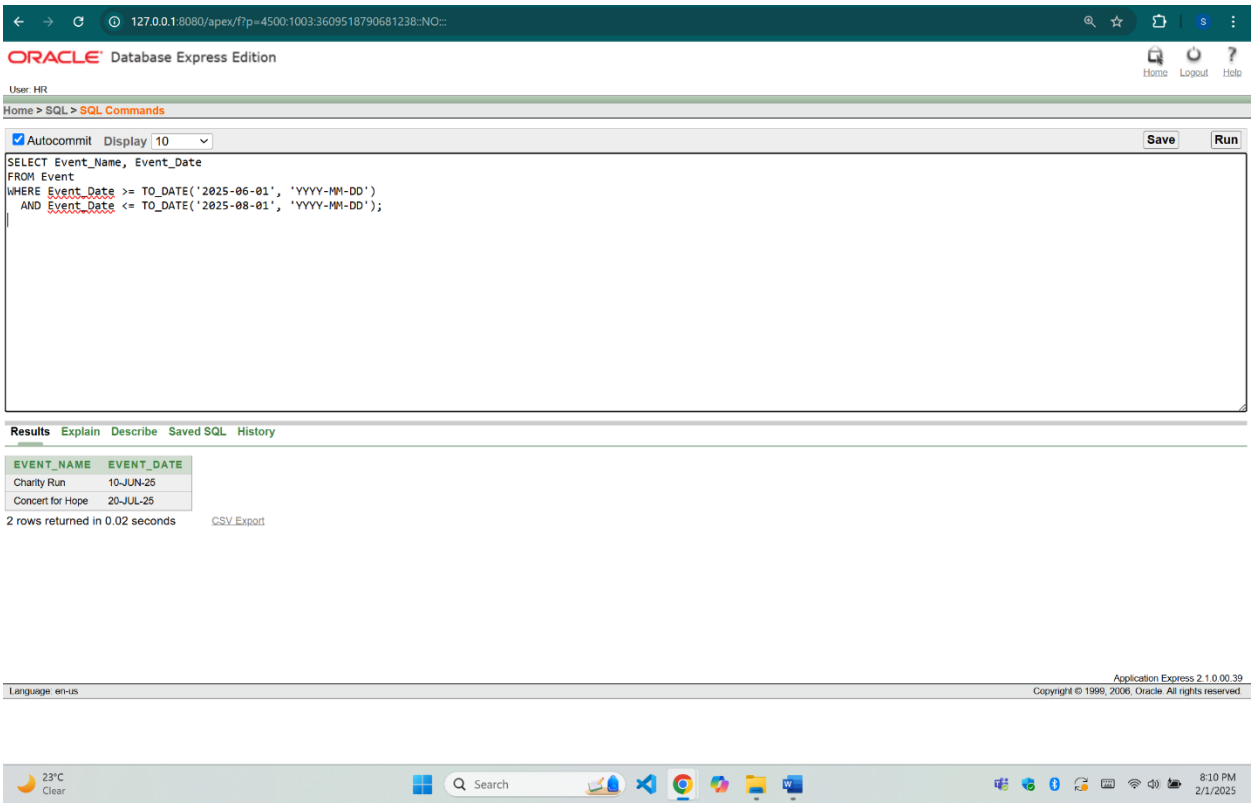


2. Operators

Query 1: (Comparison operator)

Question: What is the purpose of using the `TO_DATE` function in the WHERE clause to filter events between June 1, 2025, and August 1, 2025?

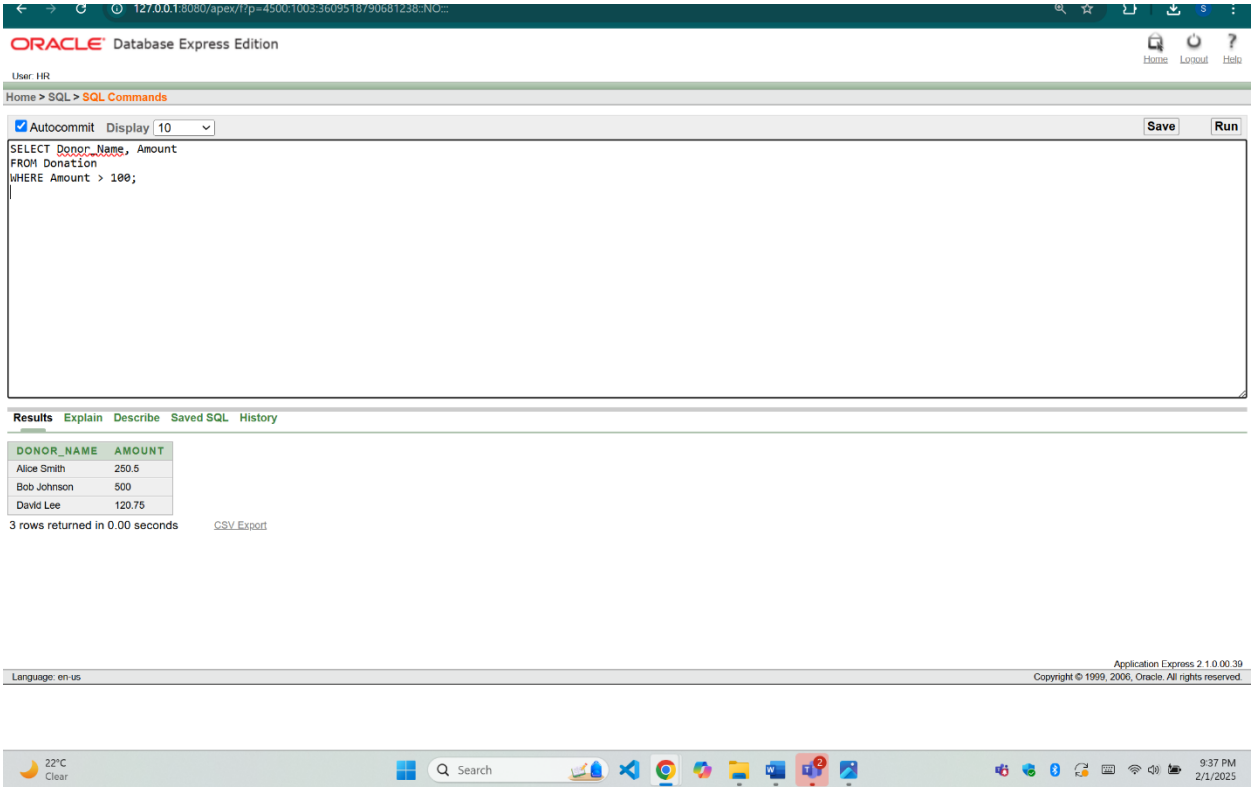
```
SELECT Event_Name, Event_Date

FROM Event

WHERE Event_Date>= TO_DATE('2025-06-01', 'YYYY-MM-DD')

  AND Event_Date<= TO_DATE('2025-08-01', 'YYYY-MM-DD');
```



Query 2: (Logical operator)

- **Question:** What is the role of the logical operator > in the WHERE clause of the second query? How does it filter the donations?

```
SELECT Donor_Name, Amount

FROM Donation

WHERE Amount > 100;
```

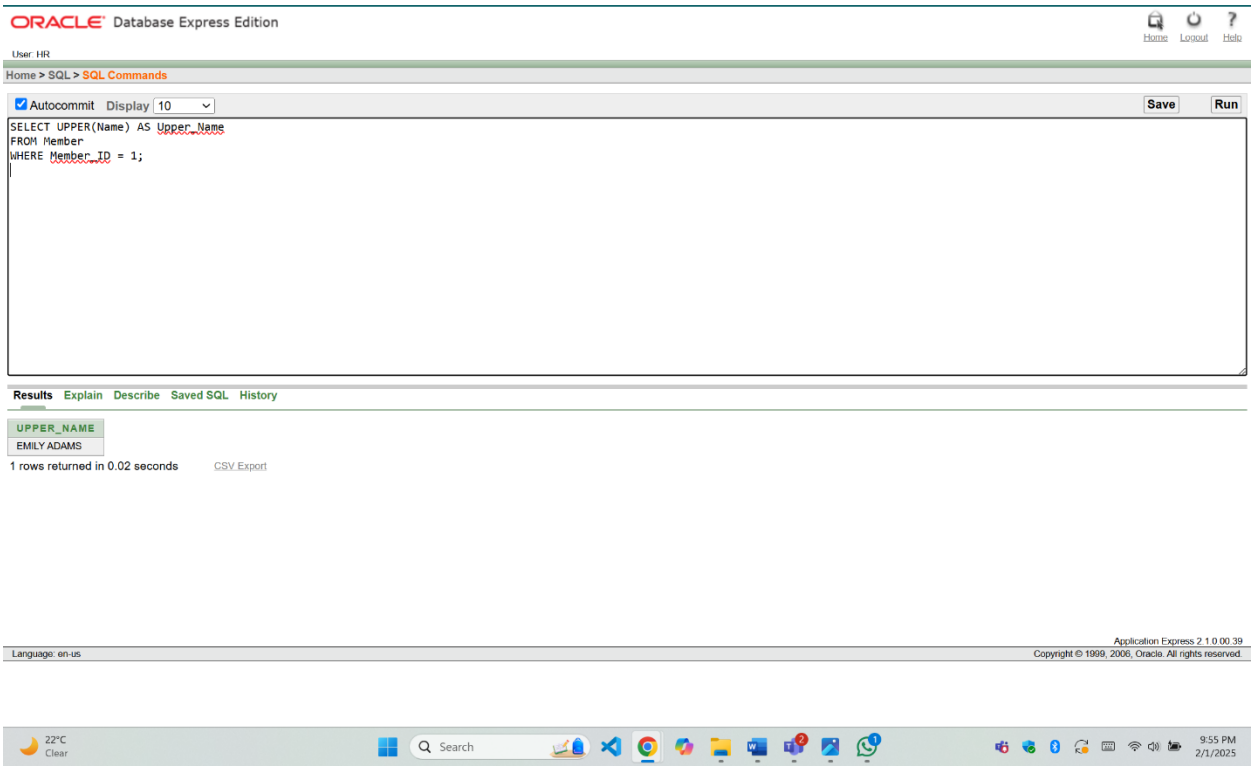


3. Single-Row Functions

Query 1: (Using `UPPER` function)

Question: How does the `UPPER` function modify the value of `Name` in the Member table for Member_ID 1?

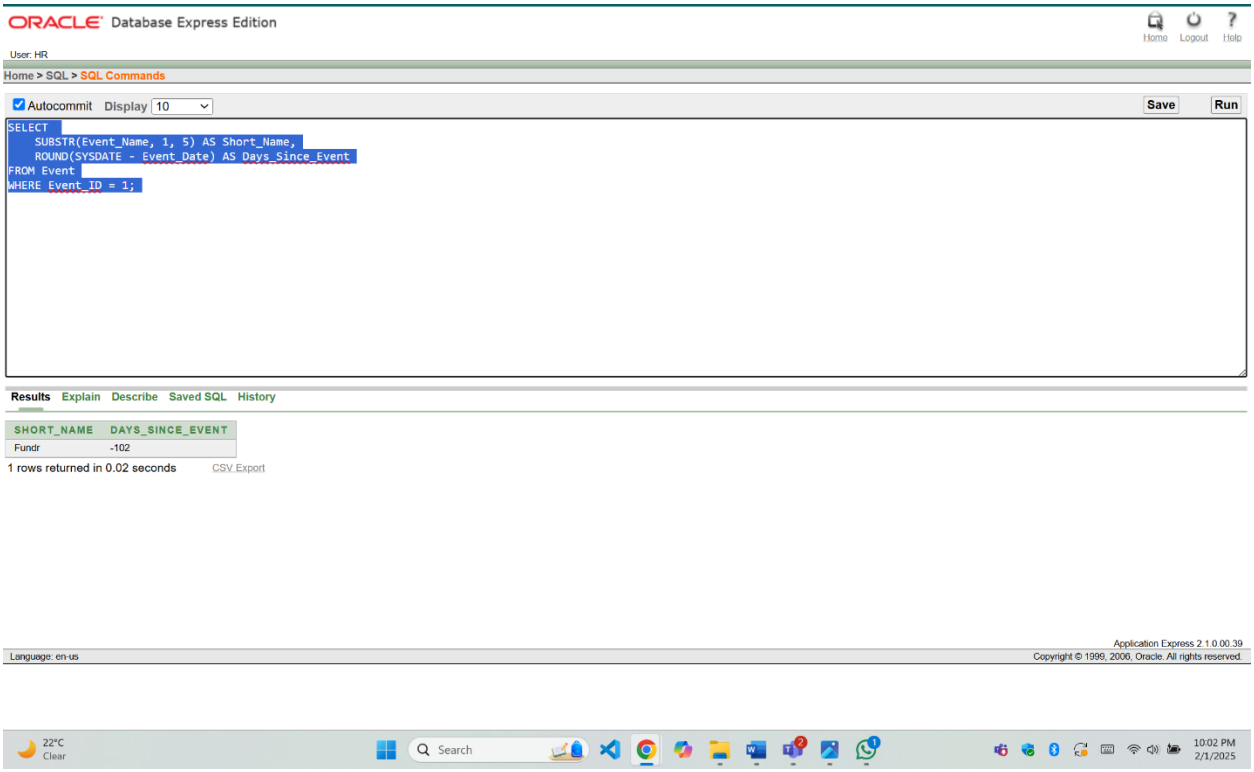
```
SELECT UPPER(Name) AS Upper_Name
FROM Member
WHERE Member_ID = 1;
```



Query 2: (Using ROUND function)

Question: What is the purpose of the `ROUND` function in calculating `Days_Since_Event` in the second query?

```
SELECT

    SUBSTR(Event_Name, 1, 5) AS Short_Name,

    ROUND(SYSDATE - Event_Date) AS Days_Since_Event

FROM Event

WHERE Event_ID = 1;
```

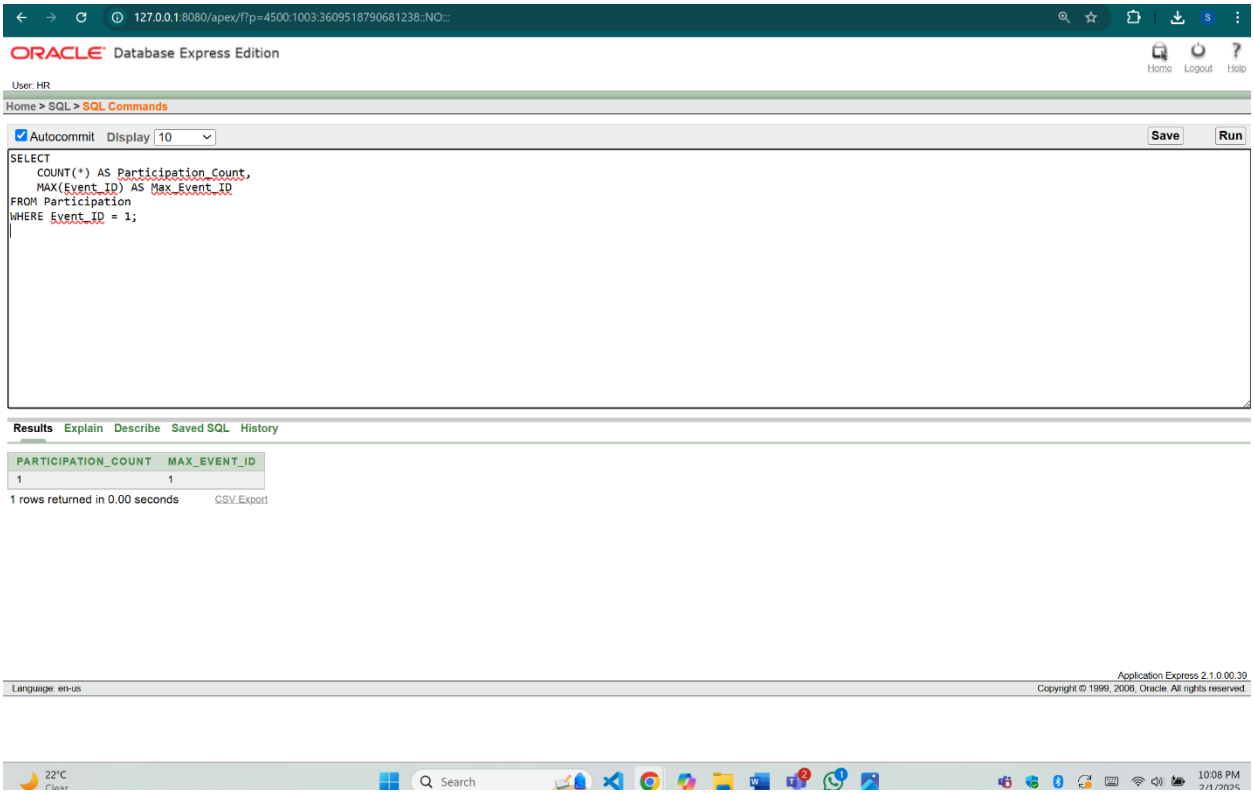


4. Group Functions

Query 1: (Using MAXCOUNT)

Question: How do the `COUNT` and `MAX` functions help analyze participation for Event_ID 1?

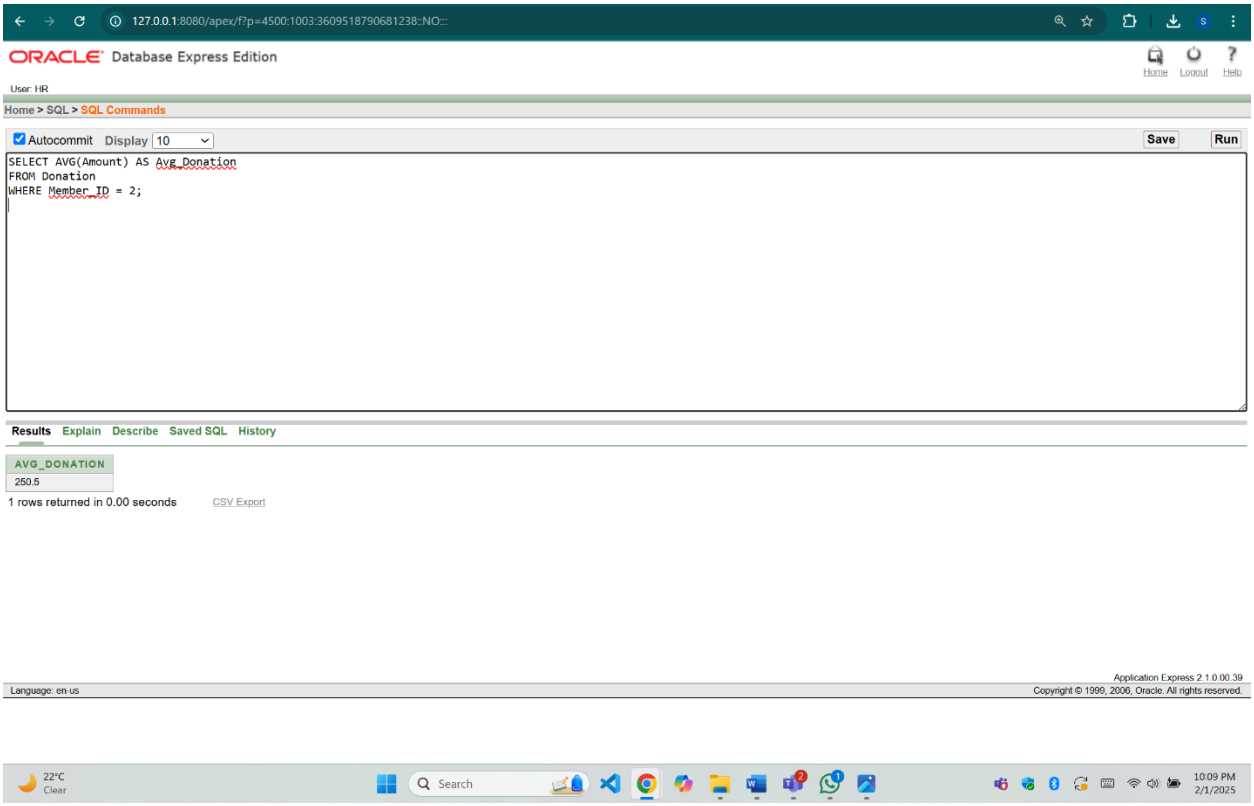
```
SELECT
    COUNT(*) AS Participation_Count,
    MAX(Event_ID) AS Max_Event_ID
FROM Participation
WHERE Event_ID = 1;
```



Query 2: (Using AVG)

Question: How does the `AVG` function calculate the average donation amount for Member_ID 2?

```
SELECT AVG(Amount) AS Avg_Donation
FROM Donation
WHERE Member_ID = 2;
```

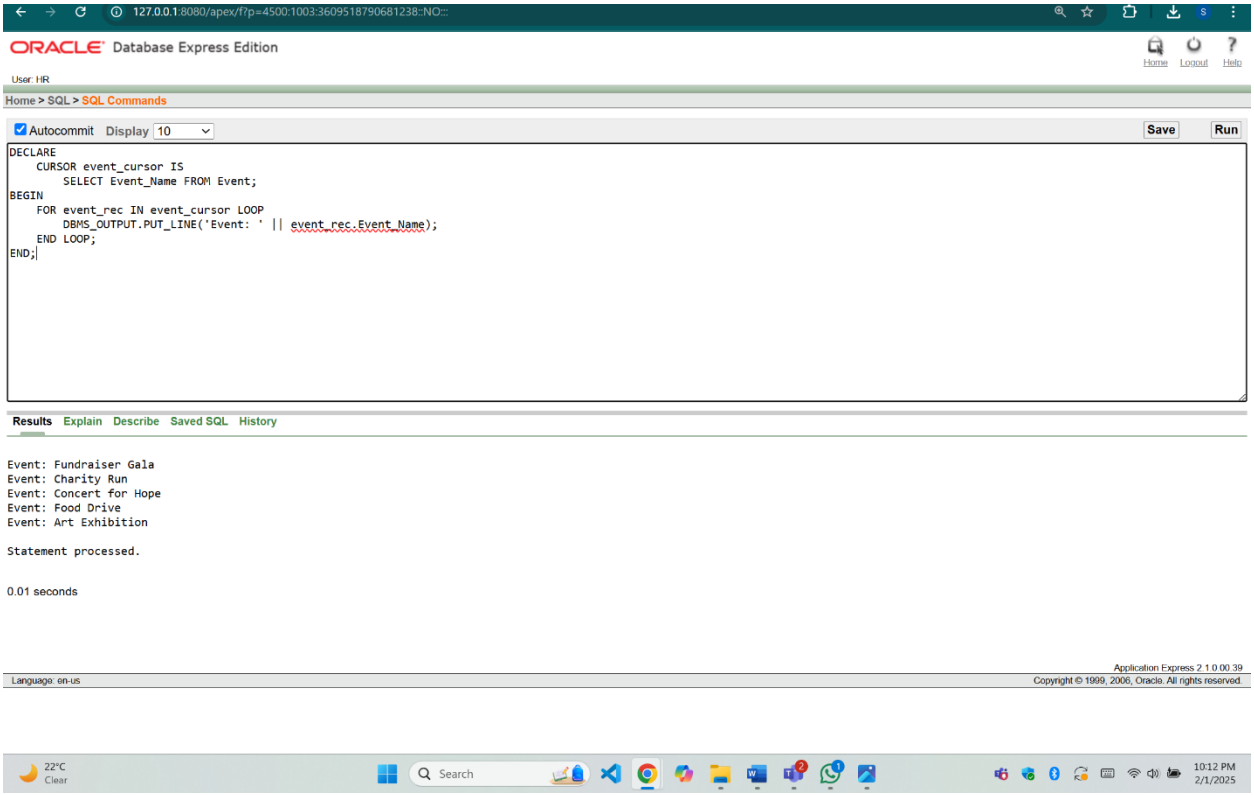


5. Loop (PL/SQL)

Query 1: (Loop through events)

Question: How does the loop fetch and display all event names using a cursor?

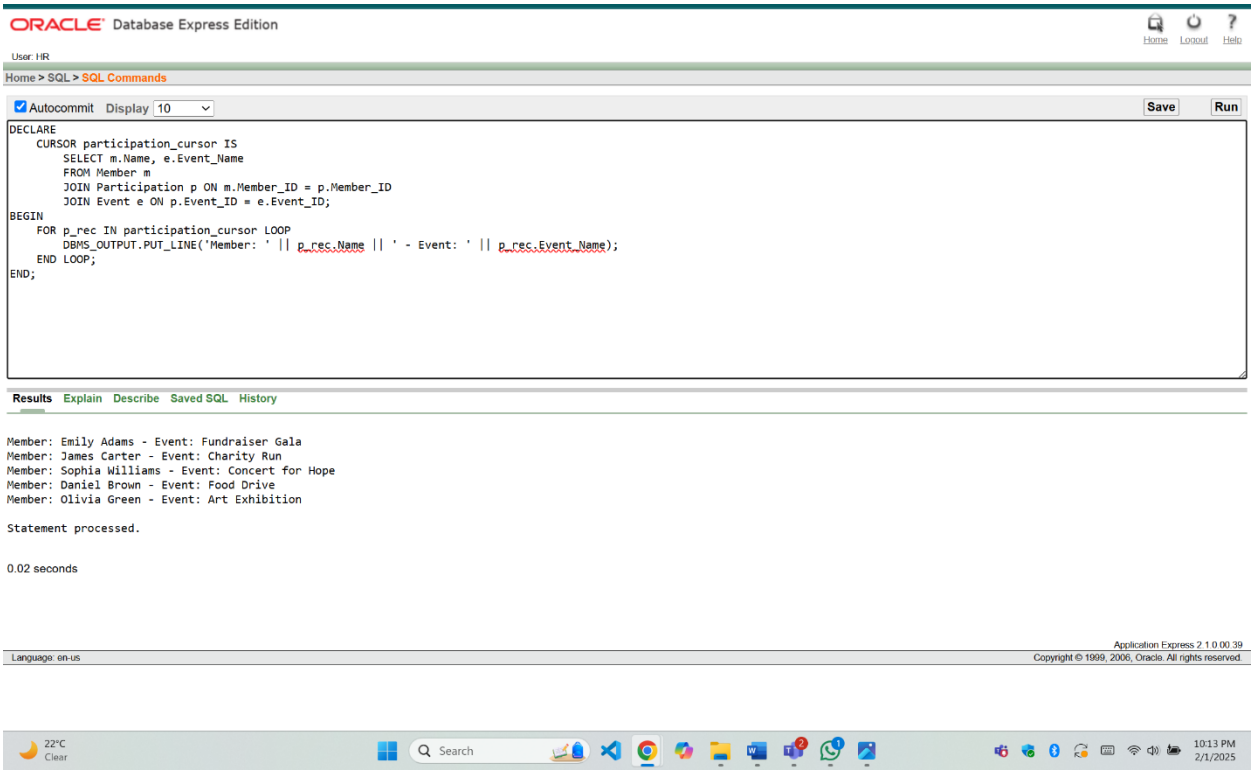
```
DECLARE
    CURSOR event_cursor IS
        SELECT Event_Name FROM Event;
BEGIN
    FOR event_rec IN event_cursor LOOP
        DBMS_OUTPUT.PUT_LINE('Event: ' || event_rec.Event_Name);
    END LOOP;
END;
```



Query 2: (Loop through participation)

Question: How does the loop through `participation_cursor` display the member's name and the event they participated in?

```
DECLARE
    CURSOR participation_cursor IS
        SELECT m.Name, e.Event_Name
        FROM Member m
        JOIN Participation p ON m.Member_ID = p.Member_ID
        JOIN Event e ON p.Event_ID = e.Event_ID;
BEGIN
    FOR p_rec IN participation_cursor LOOP
        DBMS_OUTPUT.PUT_LINE('Member: ' || p_rec.Name || ' - Event: ' ||
p_rec.Event_Name);
    END LOOP;
END;
```

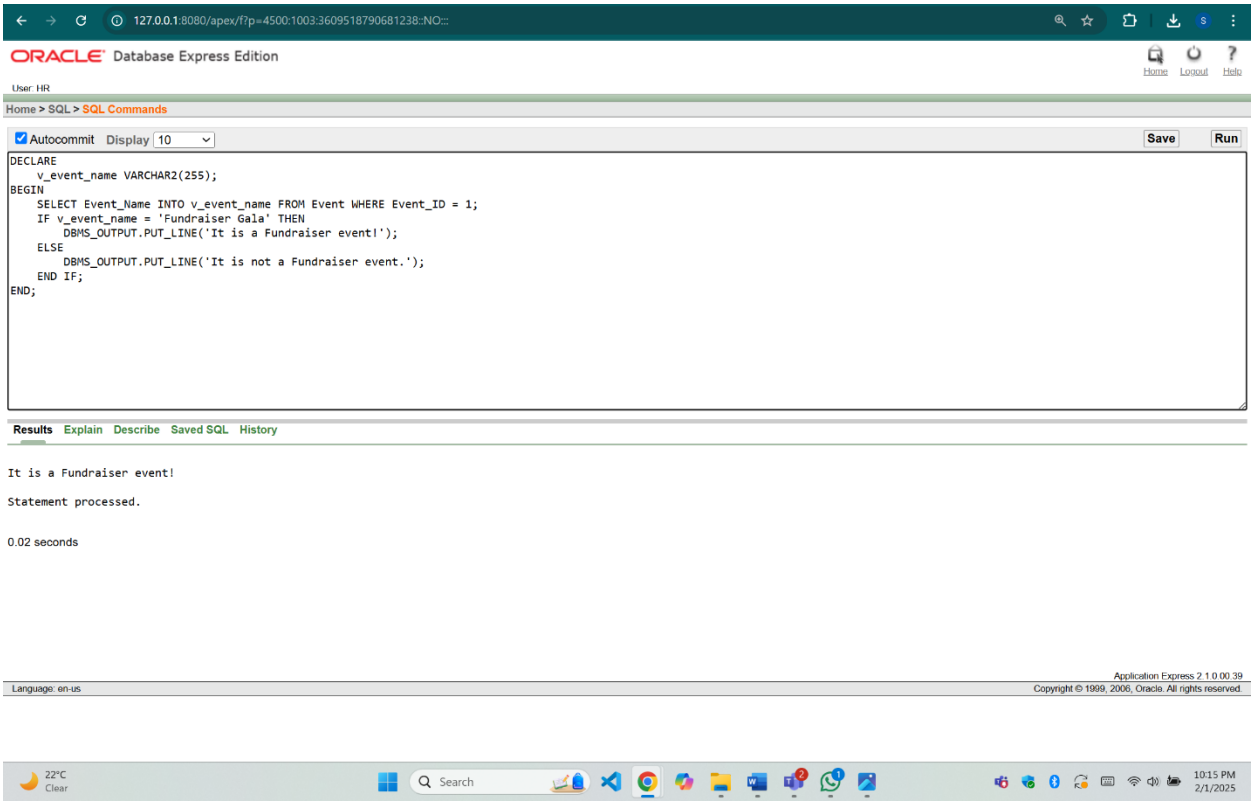


6. Conditional Statements

Query 1: (Using `IF` in PL/SQL)

Question: What happens in the `IF` condition if the event name is 'Fundraiser Gala'?

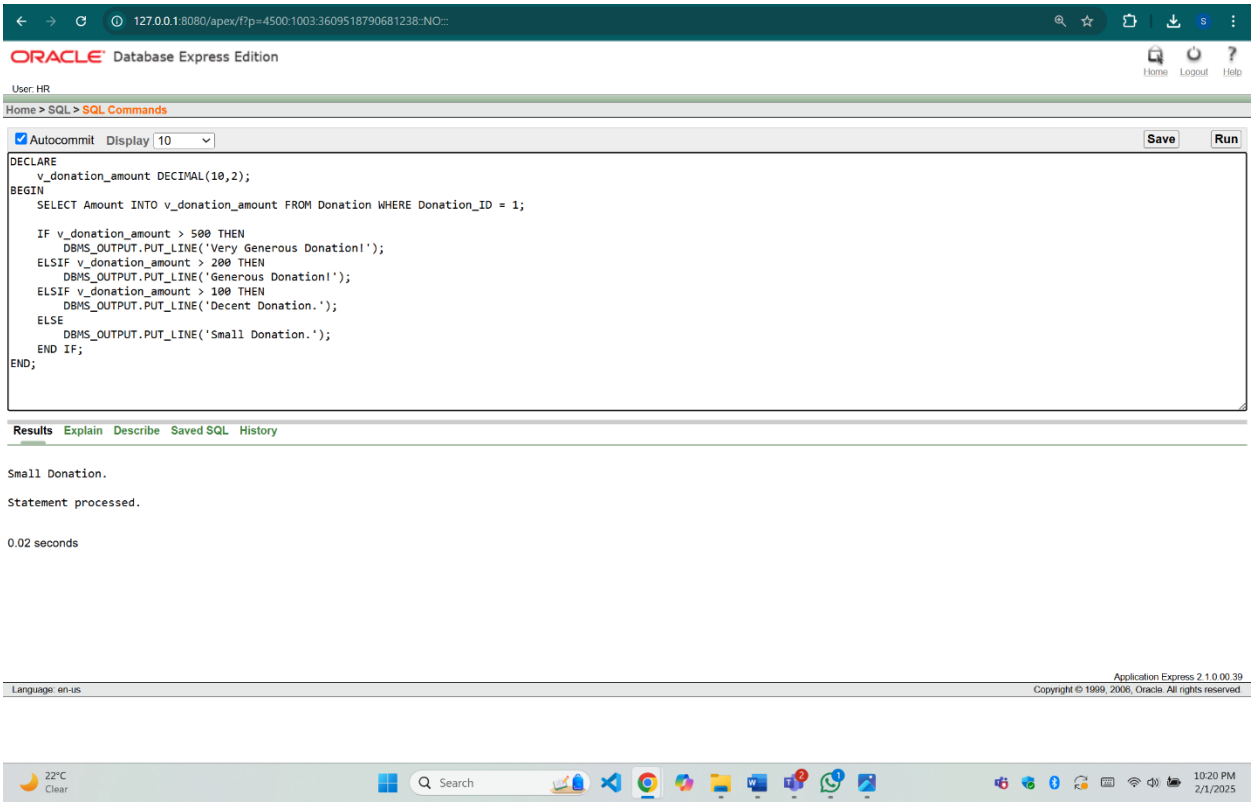
```
DECLARE
v_event_name VARCHAR2(255);
BEGIN
    SELECT Event_Name INTO v_event_name FROM Event WHERE Event_ID = 1;
    IF v_event_name = 'Fundraiser Gala' THEN
        DBMS_OUTPUT.PUT_LINE('It is a Fundraiser event!');
    ELSE
        DBMS_OUTPUT.PUT_LINE('It is not a Fundraiser event.');
    END IF;
END;
```



Query 2: (Using `ELSIF` for Multiple Conditions**)**

- **Question:** How does the `ELSIF` statement categorize the donation amount into different categories (Very Generous, Generous, Decent, Small)?

```
DECLARE
v_donation_amount DECIMAL(10,2);
BEGIN
    SELECT Amount INTO v_donation_amount FROM Donation WHERE Donation_ID = 1;

    IF v_donation_amount> 500 THEN
        DBMS_OUTPUT.PUT_LINE('Very Generous Donation!');
    ELSIF v_donation_amount> 200 THEN
        DBMS_OUTPUT.PUT_LINE('Generous Donation!');
    ELSIF v_donation_amount> 100 THEN
        DBMS_OUTPUT.PUT_LINE('Decent Donation.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Small Donation.');
    END IF;
END;
```

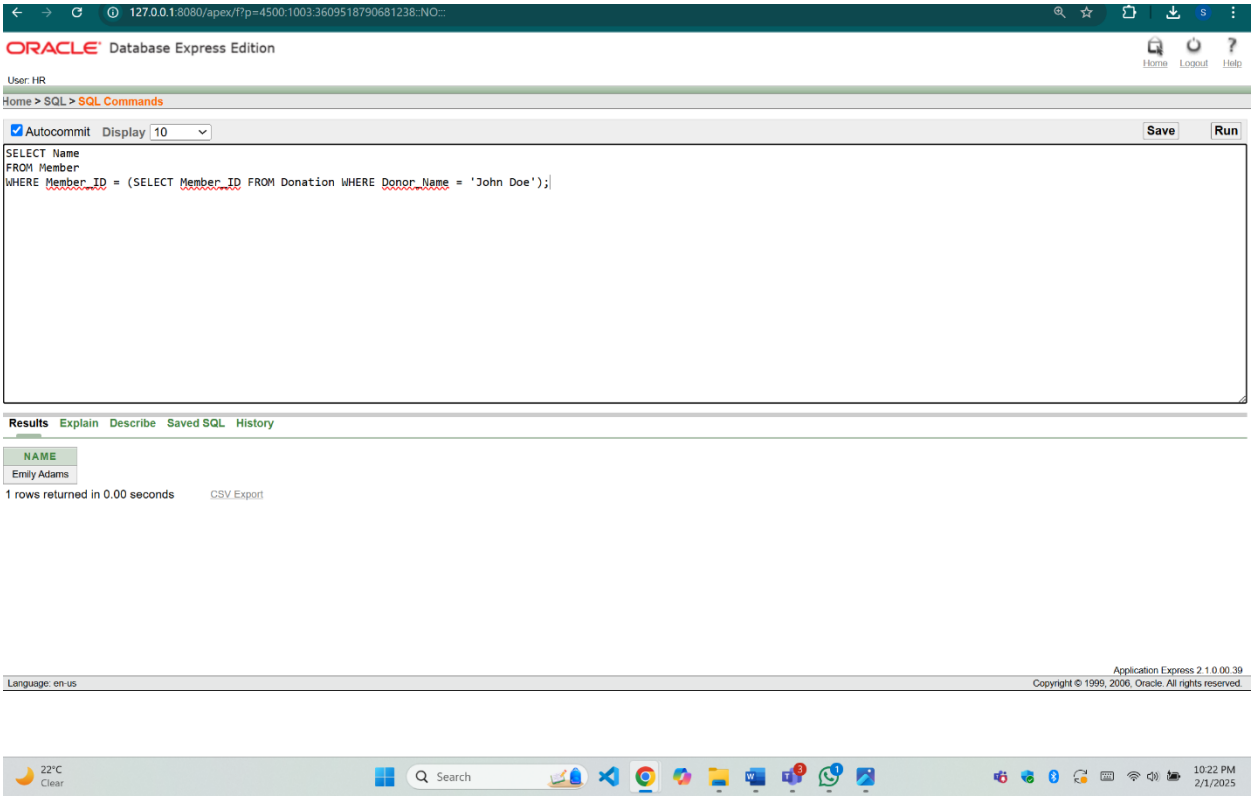


7. Subquery

Query 1: (Subquery in WHERE clause)

Question: How does the subquery in the `WHERE` clause find a member based on their donation name?

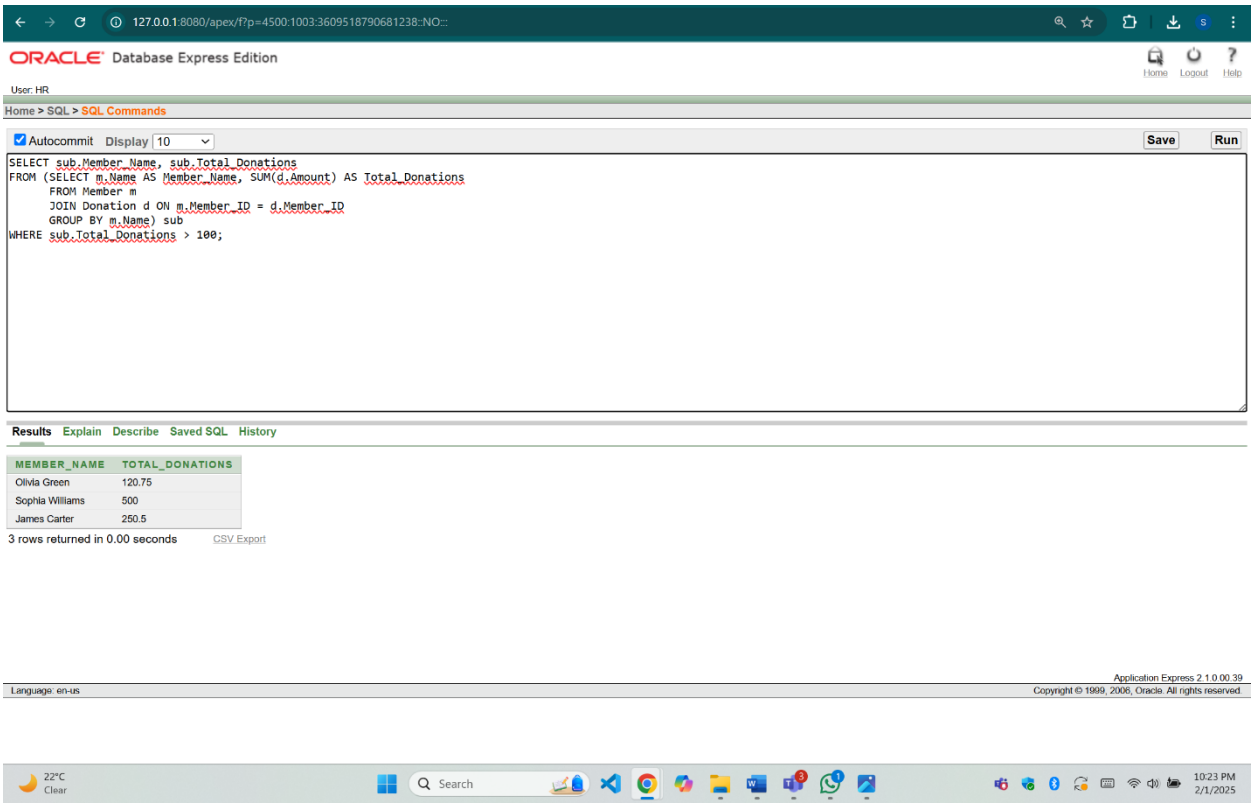
```
SELECT Name
FROM Member
WHERE Member_ID = (SELECT Member_ID FROM Donation WHERE Donor_Name = 'John
Doe');
```



Query 2: (Subquery in FROM clause)

- **Question:** How does the subquery in the `FROM` clause calculate the total donations for each member and filter those with a total above 100?

```
SELECT sub.Member_Name, sub.Total_Donations
FROM (SELECT m.Name AS Member_Name, SUM(d.Amount) AS Total_Donations
      FROM Member m
      JOIN Donation d ON m.Member_ID = d.Member_ID
      GROUP BY m.Name) sub
WHERE sub.Total_Donations> 100;
```

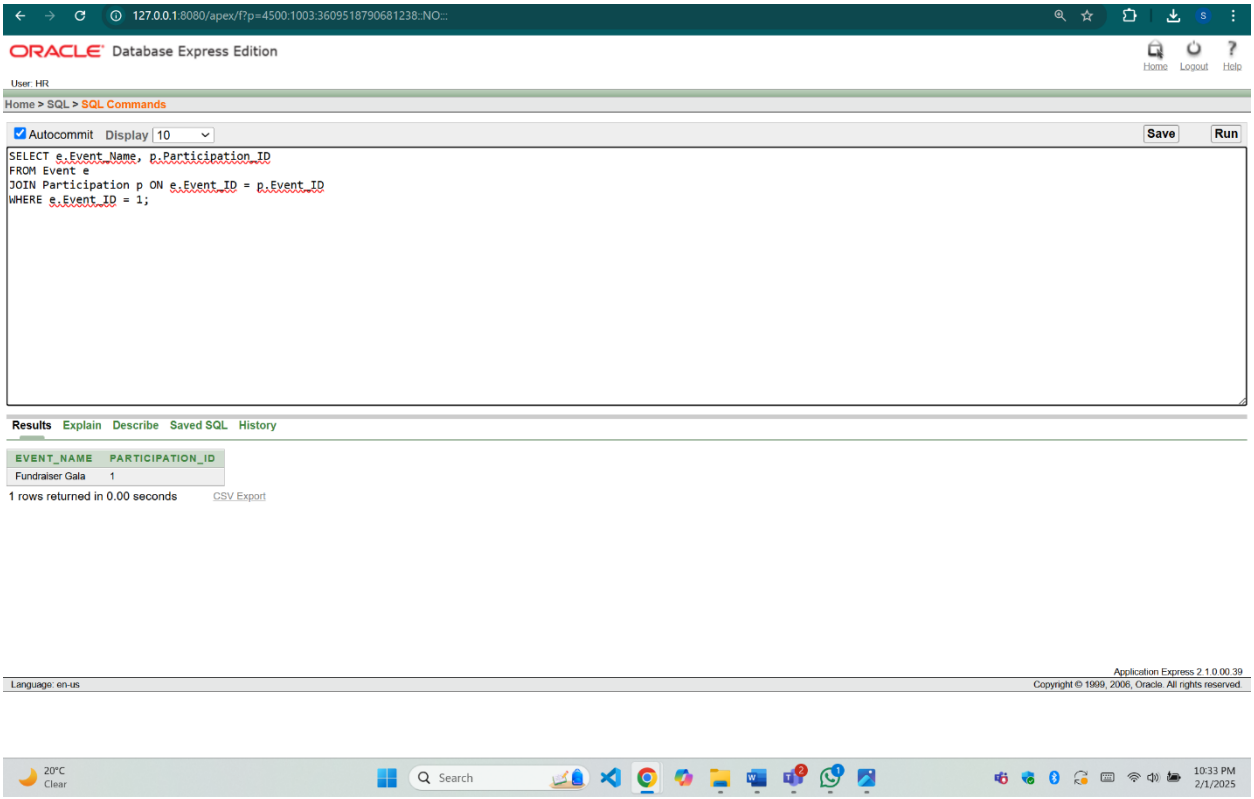


8. Joining Tables

Query 1: (Simple Join between `Event` and `Participation`)

Question: What is the result of the `JOIN` operation between the `Event` and `Participation` tables in the first query?

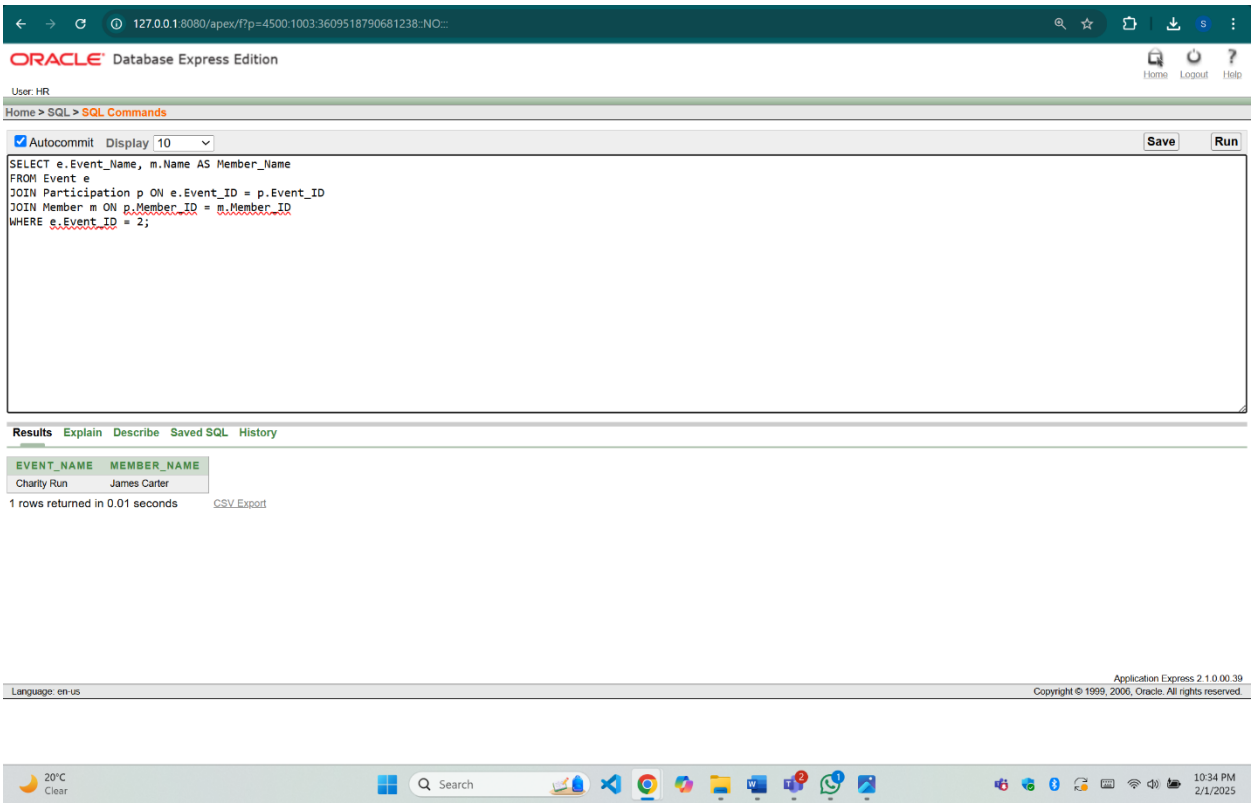
```
SELECT e.Event_Name, p.Participation_ID
FROM Event e
JOIN Participation p ON e.Event_ID = p.Event_ID
WHERE e.Event_ID = 1;
```



Query 2: (Join between `Event`, `Participation`, and `Member`)

Question: How does the second query return the names of members who participated in a specific event (Event_ID = 2)?

```
SELECT e.Event_Name, m.Name AS Member_Name
FROM Event e
JOIN Participation p ON e.Event_ID = p.Event_ID
JOIN Member m ON p.Member_ID = m.Member_ID
WHERE e.Event_ID = 2;
```



1. Function

Query 1: Create a function to calculate total donation by a member

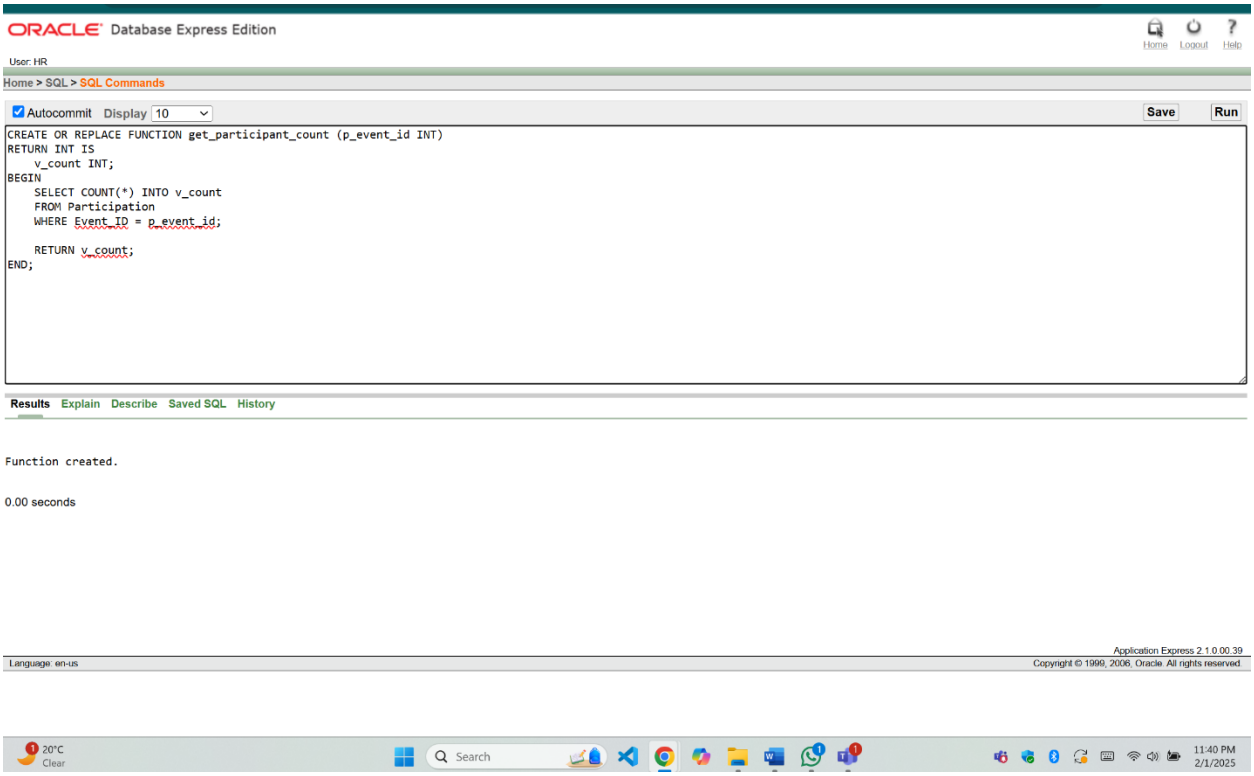
Question: **How does the `get_total_donations` function work to return the total donations made by a specific member?**

```
CREATE OR REPLACE FUNCTION get_total_donations (p_member_id INT)

RETURN DECIMAL IS

v_total DECIMAL(10, 2) := 0;

    SELECT NVL(SUM(Amount), 0) INTO v_total

    WHERE Member_ID = p_member_id;


    RETURN v_total;

END;
```

Query 2: Stored function to return the number of participants for an event

Question: **What is the purpose of the `get_participant_count` function, and how does it return the number of participants for a given event?**

```
CREATEOR REPLACE FUNCTIONget_participant_count (p_event_idINT)
RETURNINTIS
v_countINT;
BEGIN
SELECTCOUNT(*) INTOv_count
FROM Participation
WHEREEvent_ID=p_event_id;

RETURNv_count;
END;
```



This function returns the number of participants for a specific event.

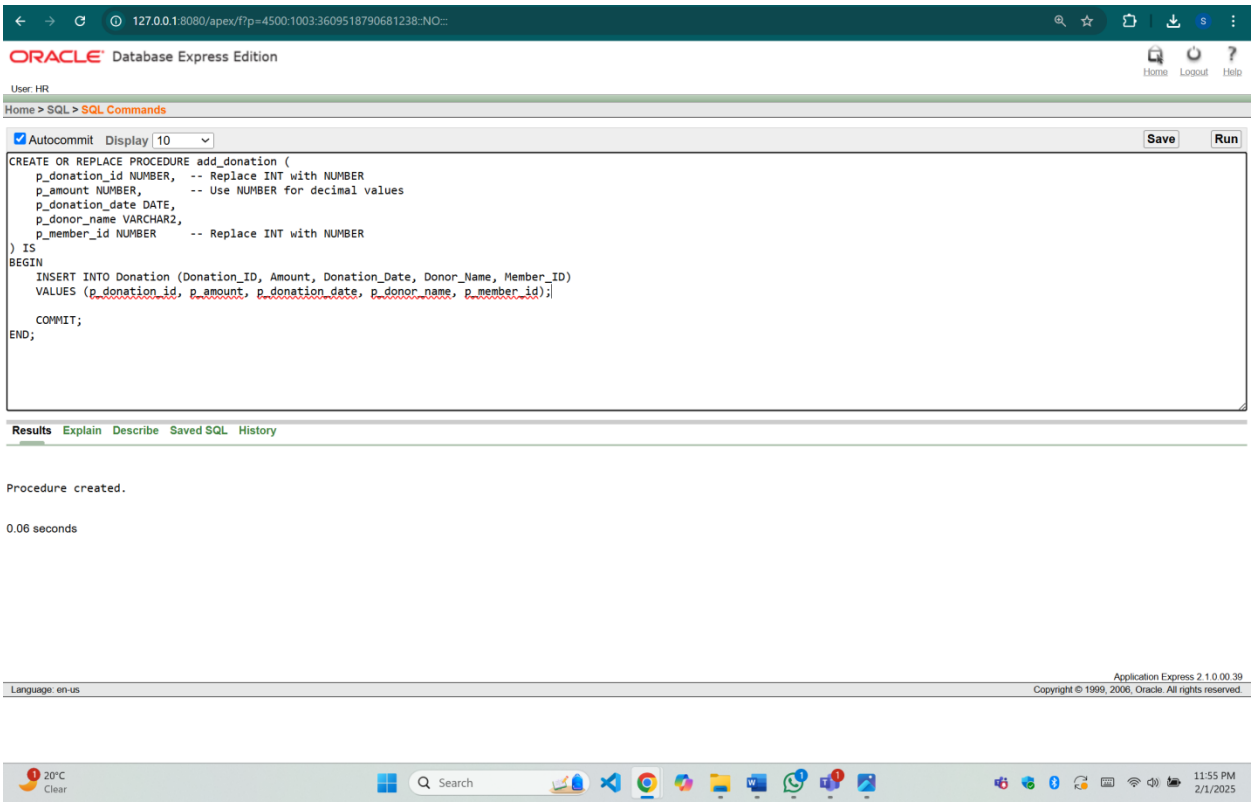
---

2. Stored Procedure

Query 1: Stored procedure to add a new donation

Question: **How does the `add_donation` stored procedure insert a new donation into the `Donation` table?**

```
CREATE OR REPLACE PROCEDURE add_donation (
p_donation_id NUMBER,  -- Replace INT with NUMBER
p_amount NUMBER,       -- Use NUMBER for decimal values
p_donation_date DATE,
p_donor_name VARCHAR2,
p_member_id NUMBER     -- Replace INT with NUMBER
) IS
BEGIN
   INSERT INTO Donation (Donation_ID, Amount, Donation_Date, Donor_Name, Member_ID)
   VALUES (p_donation_id, p_amount, p_donation_date, p_donor_name, p_member_id);

   COMMIT;
END;
```

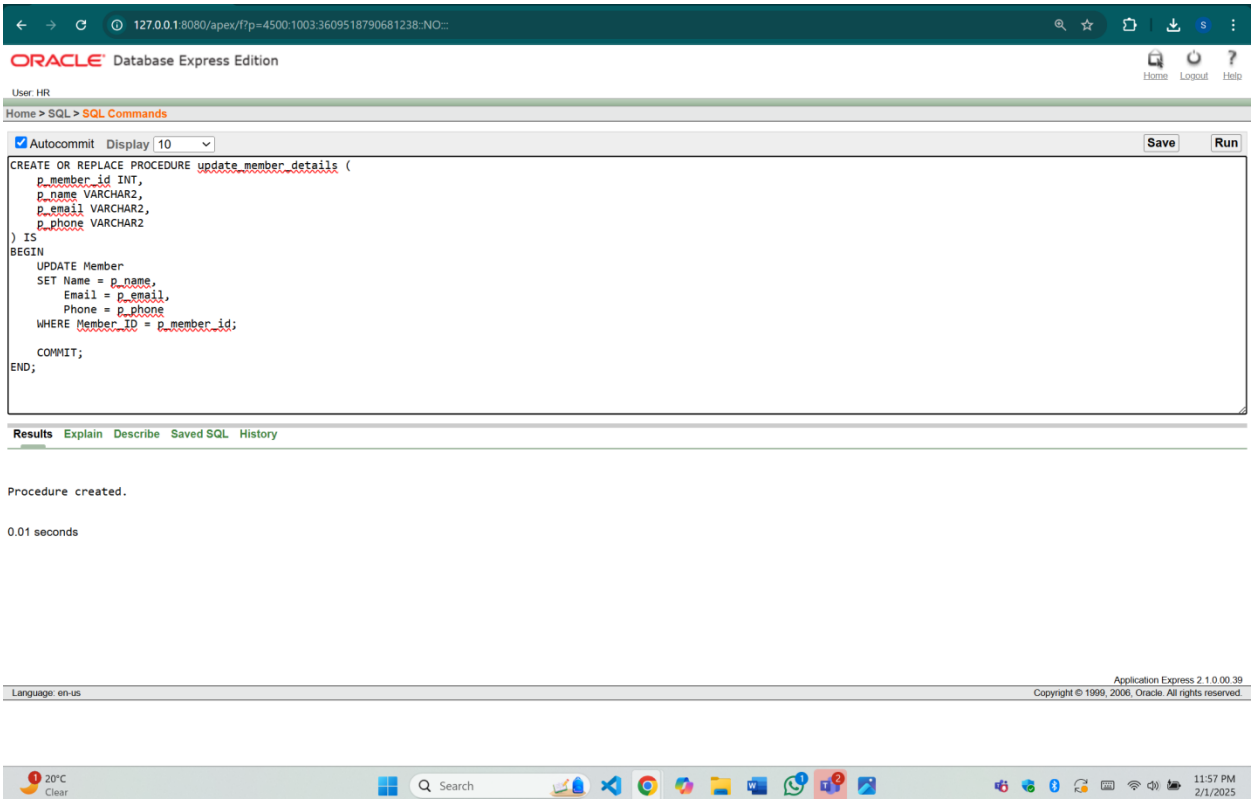


This stored procedure inserts a new donation into the `Donation` table.

Query 2: Stored procedure to update member details

- **Question:** How does the `update_member_details` stored procedure update a member's name, email, and phone?

```
CREATEOR REPLACE PROCEDUREupdate_member_details (
p_member_idINT,
p_name VARCHAR2,
p_email VARCHAR2,
p_phone VARCHAR2
) IS
BEGIN
UPDATEMember
SET Name =p_name,
        Email =p_email,
        Phone =p_phone
WHEREMember_ID=p_member_id;

COMMIT;
END;
```



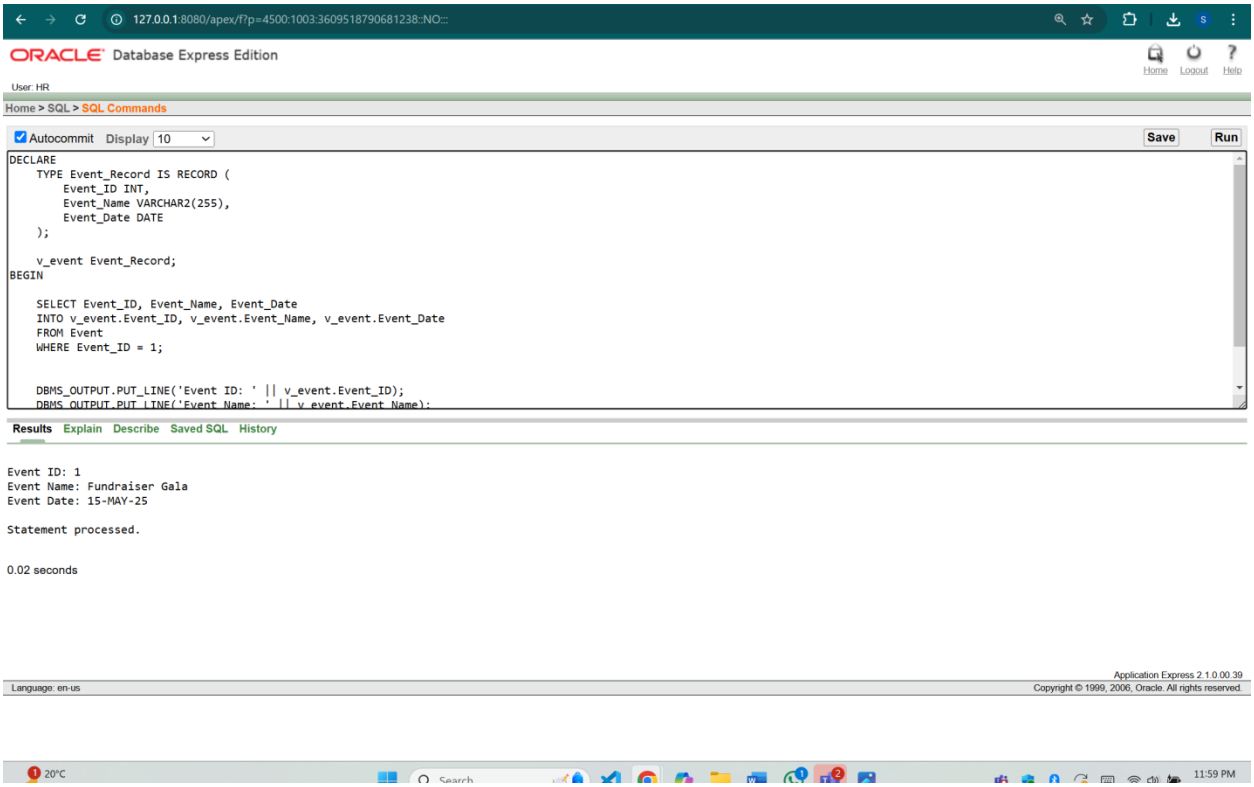
This stored procedure updates the details of a member based on `Member_ID`.

3. Table-Based Record

Query 1: Creating a record type for storing event details

Question: **How does the `Event_Record` type store event details, and how are these details retrieved and displayed in the `DBMS_OUTPUT`?**

```
DECLARE
    TYPE Event_RecordIS RECORD (
Event_IDINT,
Event_Name VARCHAR2(255),
Event_DateDATE
    );

v_eventEvent_Record;
BEGIN

SELECTEvent_ID, Event_Name, Event_Date
INTOv_event.Event_ID, v_event.Event_Name, v_event.Event_Date
FROM Event
WHEREEvent_ID=1;


    DBMS_OUTPUT.PUT_LINE('Event ID: '||v_event.Event_ID);
    DBMS_OUTPUT.PUT_LINE('Event Name: '||v_event.Event_Name);
    DBMS_OUTPUT.PUT_LINE('Event Date: '||v_event.Event_Date);
END;
```

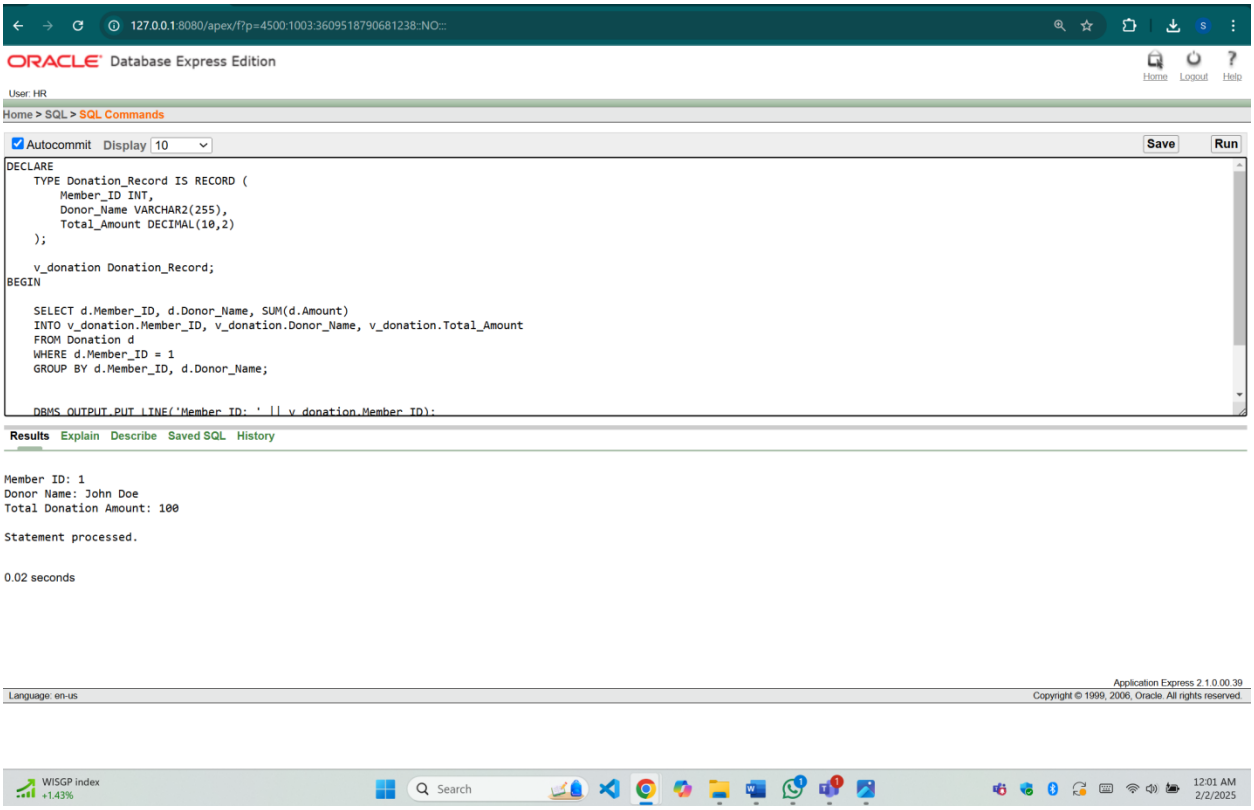


Query 2: Creating a table-based record for member donation information

Question: **How does the `Donation_Record` type store member donation information, and how is it used to display the total donations for a member?**

```
DECLARE
    TYPE Donation_RecordIS RECORD (
Member_IDINT,
Donor_Name VARCHAR2(255),
```

```
Total_AmountDECIMAL(10,2)
    );

v_donationDonation_Record;
BEGIN

SELECTd.Member_ID, d.Donor_Name, SUM(d.Amount)
INTOv_donation.Member_ID, v_donation.Donor_Name, v_donation.Total_Amount
FROM Donation d
WHEREd.Member_ID=1
GROUPBYd.Member_ID, d.Donor_Name;


    DBMS_OUTPUT.PUT_LINE('Member ID: '||v_donation.Member_ID);
    DBMS_OUTPUT.PUT_LINE('Donor Name: '||v_donation.Donor_Name);
    DBMS_OUTPUT.PUT_LINE('Total Donation Amount: '||v_donation.Total_Amount);
END;
```



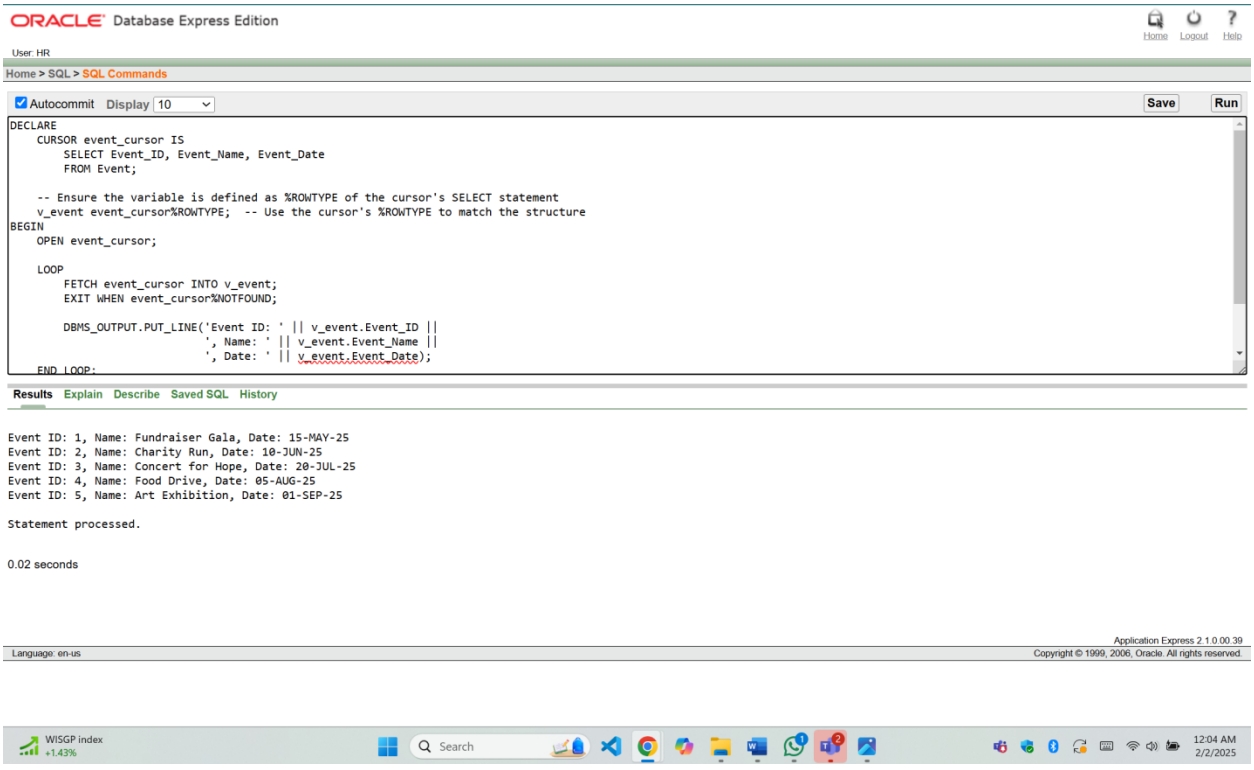
4. Explicit Cursor

Query 1: Using explicit cursor to loop through events

Question: **How does the explicit cursor `event_cursor` work to fetch event details and loop through them?**

```
CURSOR event_cursor IS
    SELECT Event_ID, Event_Name, Event_Date
```

```
    FROM Event;

  -- Ensure the variable is defined as %ROWTYPE of the cursor's SELECT statement
  v_event event_cursor%ROWTYPE;  -- Use the cursor's %ROWTYPE to match the structure
BEGIN
  OPEN event_cursor;

  LOOP
    FETCH event_cursor INTO v_event;
    EXIT WHEN event_cursor%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE('Event ID: ' || v_event.Event_ID ||
                         ', Name: ' || v_event.Event_Name ||
                         ', Date: ' || v_event.Event_Date);
  END LOOP;

  CLOSE event_cursor;
END;
```

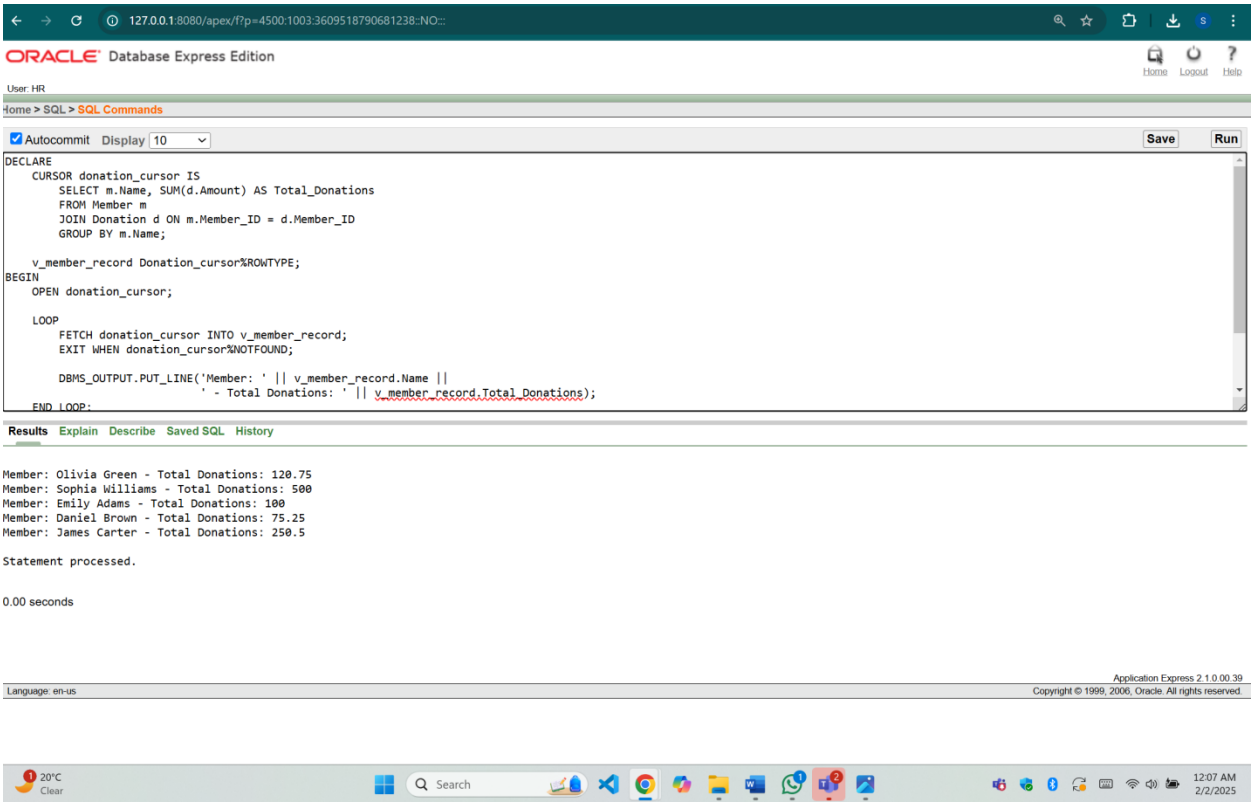


Query 2: Explicit cursor for fetching members with donations

Question: **How does the `donation_cursor` loop through member donations and display the total donations?**

```
DECLARE
CURSORdonation_cursorIS
```

```
SELECTm.Name, SUM(d.Amount) ASTotal_Donations
FROMMember m
JOIN Donation d ONm.Member_ID=d.Member_ID
GROUPBYm.Name;

v_member_recordDonation_cursor%ROWTYPE;
BEGIN
OPENdonation_cursor;

    LOOP
FETCHdonation_cursorINTOv_member_record;
        EXIT WHENdonation_cursor%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('Member: '||v_member_record.Name||
' - Total Donations: '||v_member_record.Total_Donations);
END LOOP;


CLOSEdonation_cursor;
END;
```

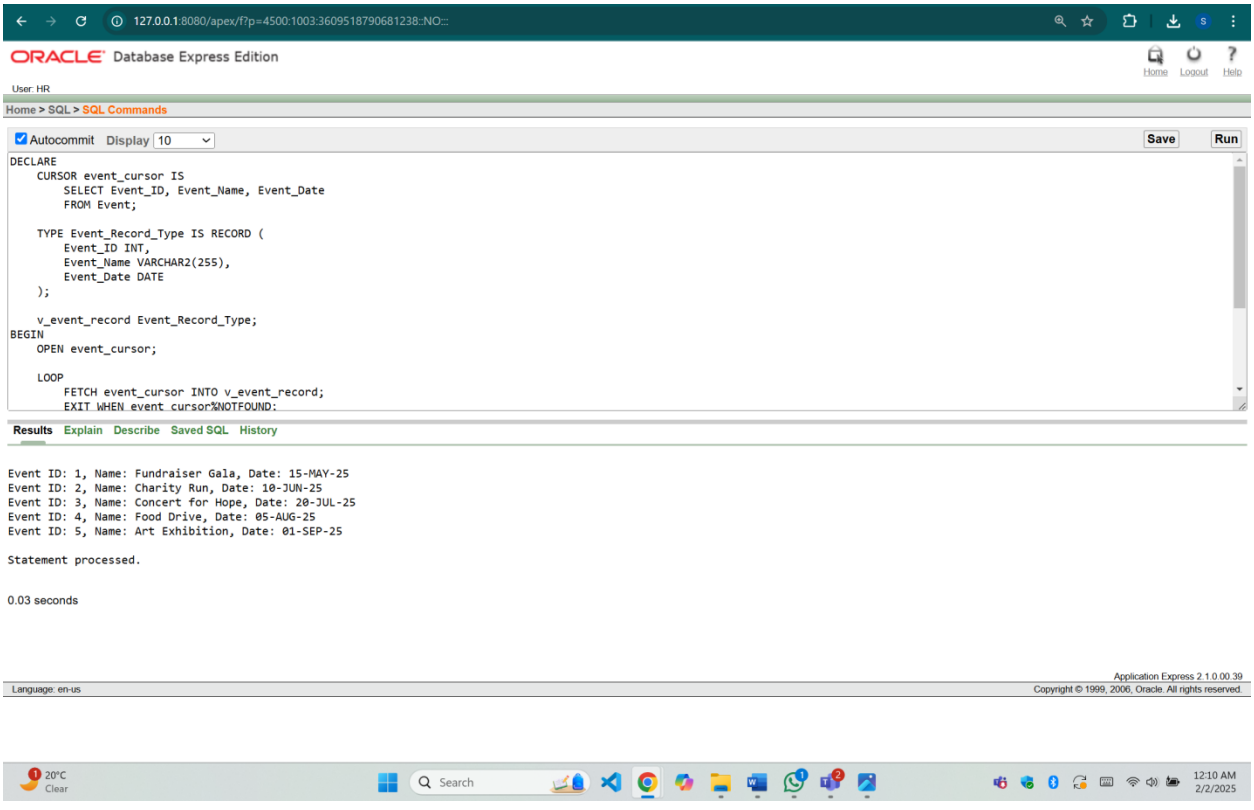


5. cursor-based Record

Query 1: Using a cursor-based record to loop through event details

Question: **What is the benefit of using a cursor-based record for looping through event data and displaying it?**

```
DECLARE
CURSOREvent_cursorIS
SELECTEvent_ID, Event_Name, Event_Date
FROM Event;

    TYPE Event_Record_TypeIS RECORD (
Event_IDINT,
Event_Name VARCHAR2(255),
Event_DateDATE
    );

v_event_recordEvent_Record_Type;
BEGIN
OPENevent_cursor;

    LOOP
FETCHevent_cursorINTOv_event_record;
        EXIT WHENevent_cursor%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('Event ID: '||v_event_record.Event_ID||
', Name: '||v_event_record.Event_Name||
', Date: '||v_event_record.Event_Date);
END LOOP;

CLOSEevent_cursor;
END;
```

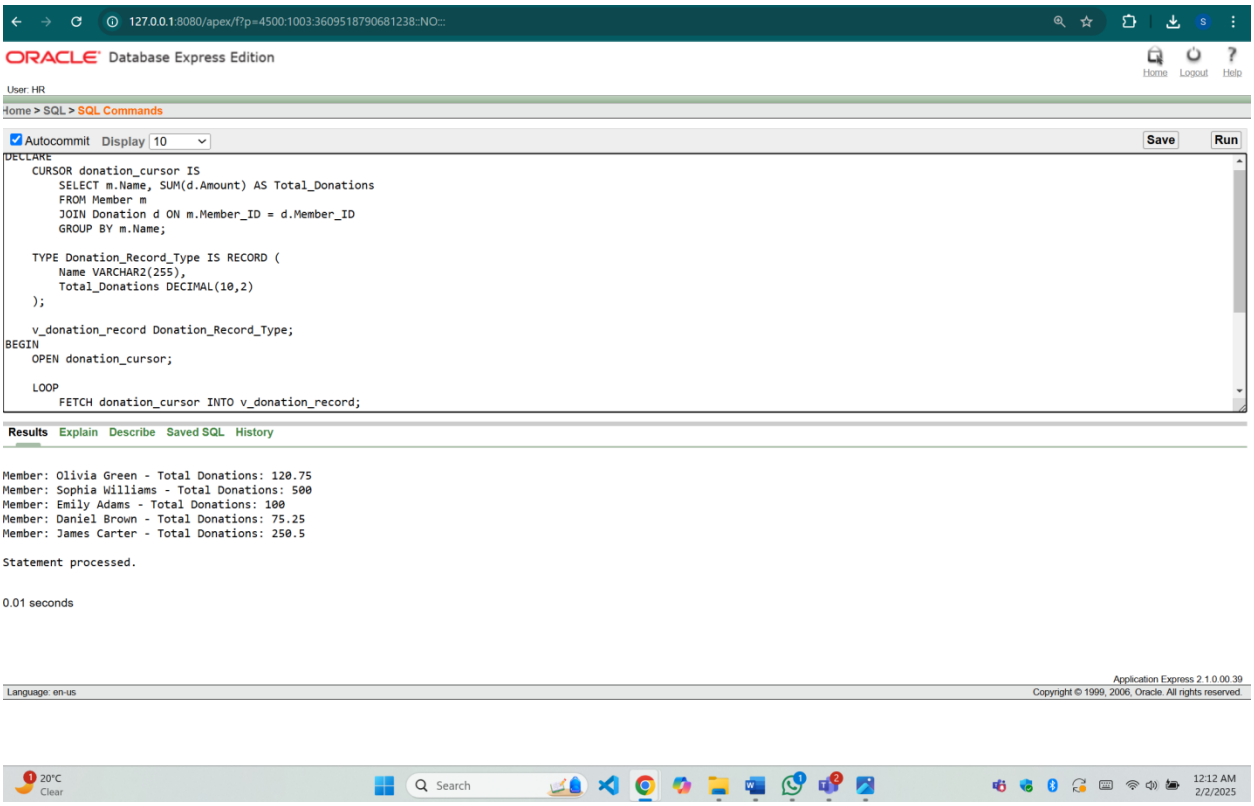


Query 2: Cursor-based record to display member donations

Question: **How does the cursor-based record in the `donation_cursor` fetch and display the total donations for each member?**

```
DECLARE
CURSORdonation_cursorIS
SELECTm.Name, SUM(d.Amount) ASTotal_Donations
FROMMember m
JOIN Donation d ONm.Member_ID=d.Member_ID
GROUPBYm.Name;

    TYPE Donation_Record_TypeIS RECORD (
        Name VARCHAR2(255),
Total_DonationsDECIMAL(10,2)
    );

v_donation_recordDonation_Record_Type;
BEGIN
OPENdonation_cursor;

    LOOP
FETCHdonation_cursorINTOv_donation_record;
        EXIT WHENdonation_cursor%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('Member: '||v_donation_record.Name||
' - Total Donations: '||v_donation_record.Total_Donations);
END LOOP;

CLOSEdonation_cursor;
```

```
END;
```



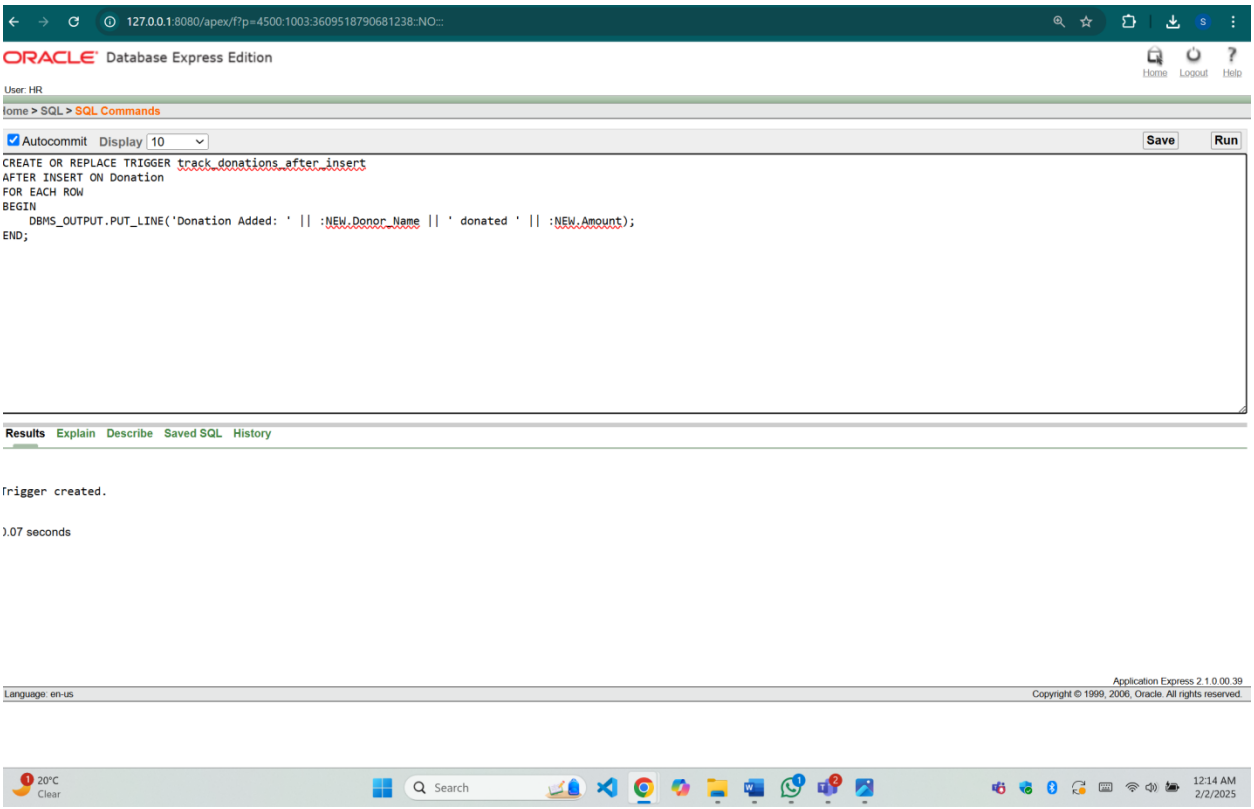
6.Row-level Trigger

Query 1: Row-level trigger to track when a donation is added

Question: **How does the `track_donations_after_insert` trigger track and display donation information after an insert operation?**

```
CREATEOR REPLACE TRIGGERtrack_donations_after_insert
AFTER INSERTON Donation
FOREACHROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('Donation Added: '|| :NEW.Donor_Name||' donated '||
:NEW.Amount);
```

```
END;
```

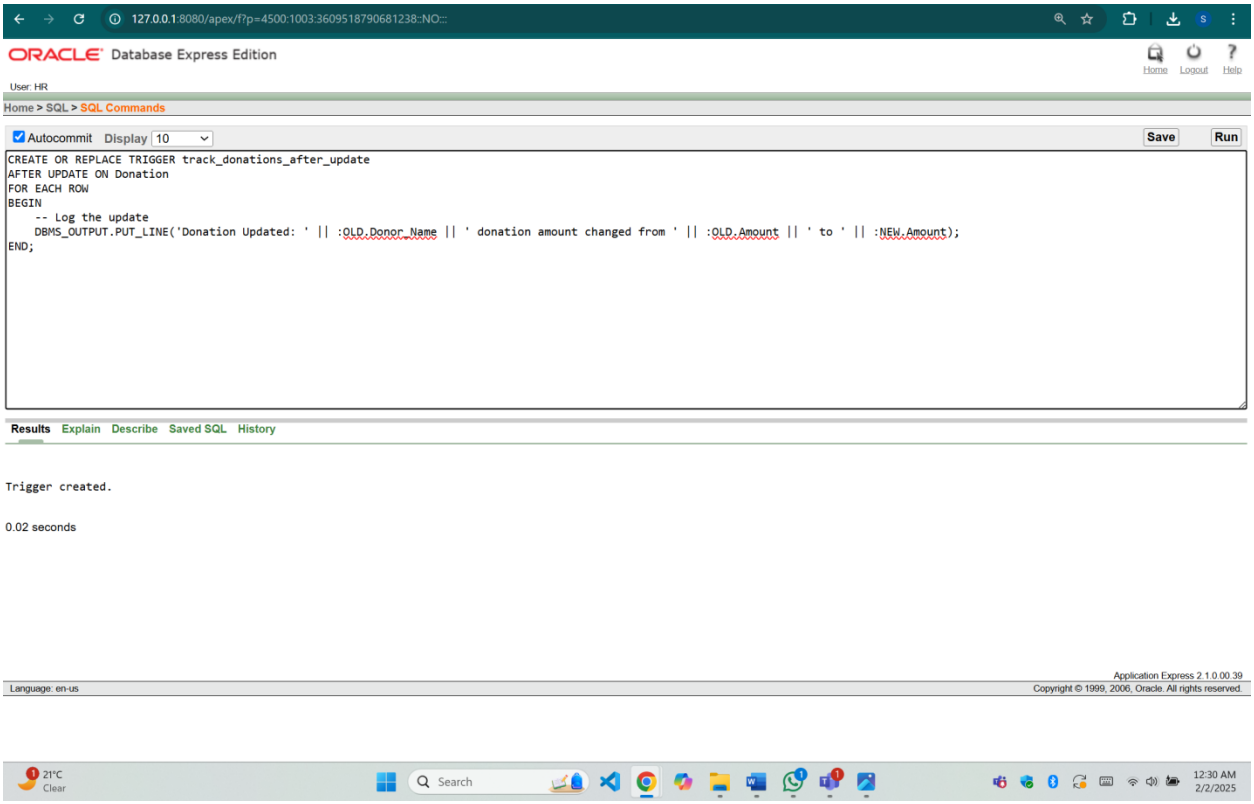


Query 2: Row-level AFTER UPDATE Trigger

- **Question:** What happens when the `track_donations_after_update` trigger is fired, and what is displayed?

```
CREATE OR REPLACE TRIGGER track_donations_after_update

AFTER UPDATE ON Donation

FOR EACH ROW

BEGIN


    DBMS_OUTPUT.PUT_LINE('Donation Updated: ' || :OLD.Donor_Name || '
donation amount changed from ' || :OLD.Amount || ' to ' || :NEW.Amount);
```

```
END;
```



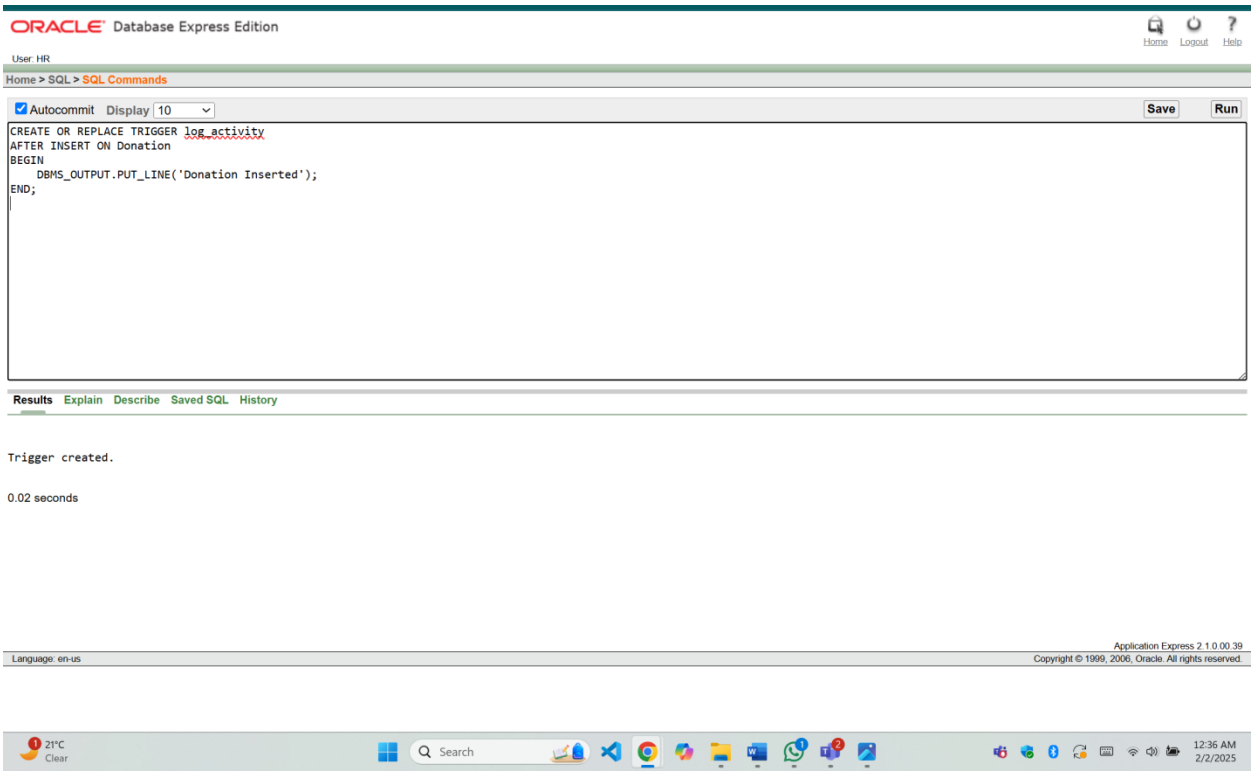
7. Statement-Level Trigger

Query 1: Statement-level trigger to log activity

Question: **How does the `log_activity` trigger log the insertion of a donation?**

```
CREATEOR REPLACE TRIGGERlog_activity
AFTER INSERTON Donation
BEGIN
    DBMS_OUTPUT.PUT_LINE('Donation Inserted');
```

```
END;
```

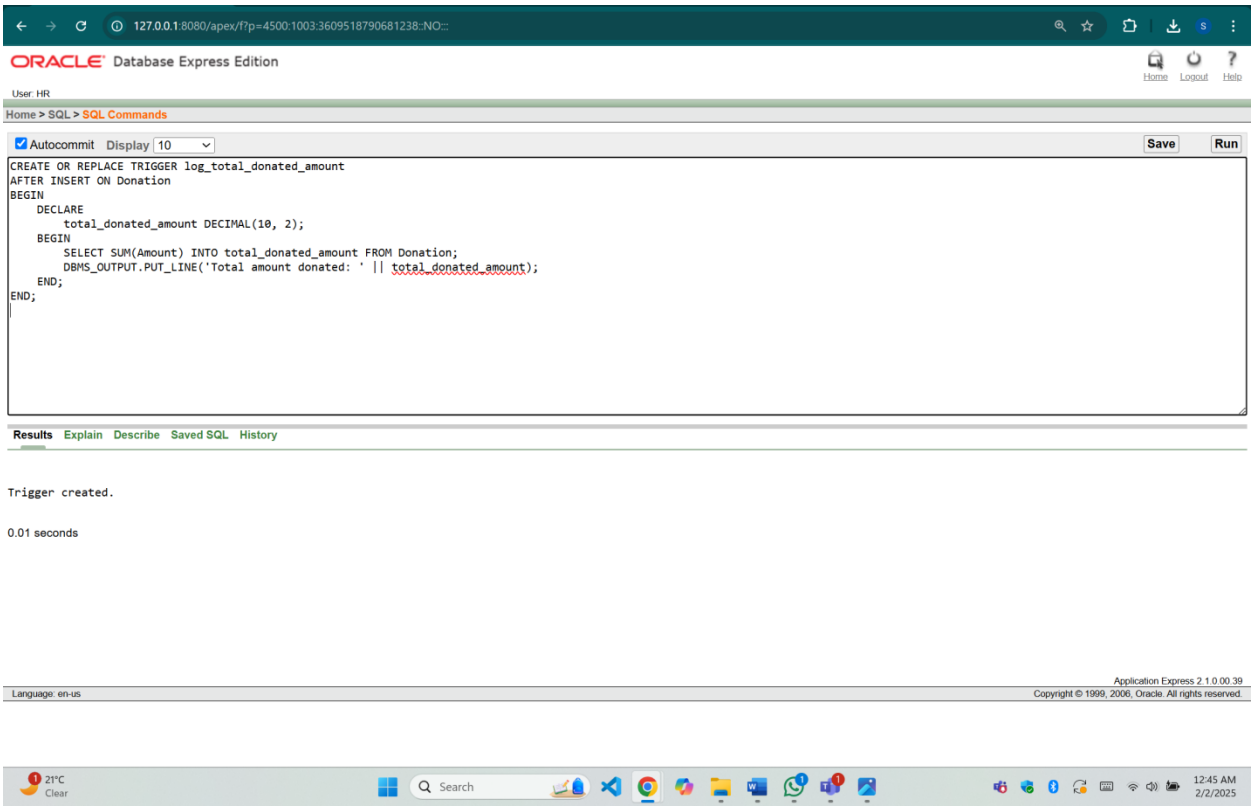


Query 2: Statement-Level Trigger to Log the Total Amount of Donations Inserted

Question: How does the `log_total_donated_amount` trigger calculate and log the total amount of donations after an insert operation?.

```
CREATE OR REPLACE TRIGGER log_total_donated_amount

AFTER INSERT ON Donation

BEGIN

    DECLARE

total_donated_amount DECIMAL(10, 2);

    BEGIN

        SELECT SUM(Amount) INTO total_donated_amount FROM Donation;

        DBMS_OUTPUT.PUT_LINE('Total amount donated: ' ||
total_donated_amount);

    END;
```

```
END;
```

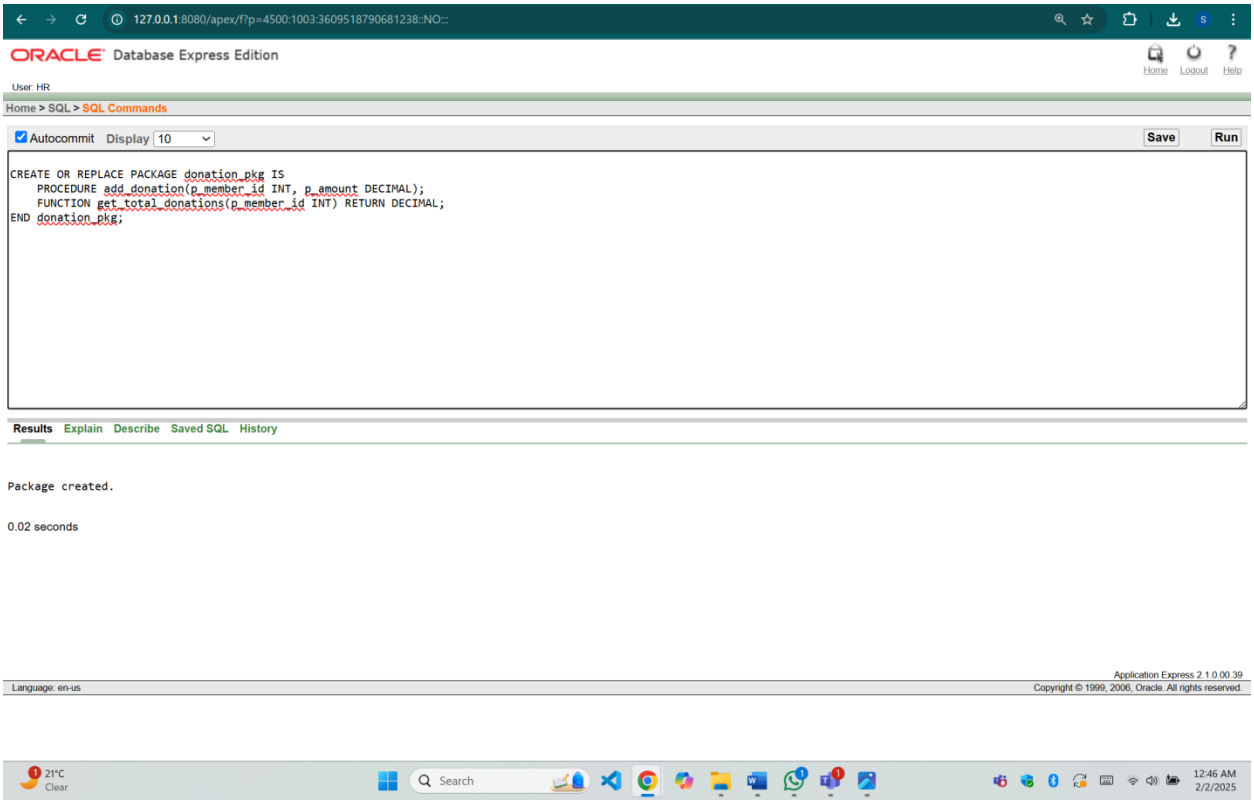


8. Package

Query 1: Create a package for managing donation-related operations

Question: **How is the package `donation_pkg` used to manage donation-related operations, and what procedures and functions does it include?**

```
CREATEOR REPLACE PACKAGE donation_pkgIS
PROCEDUREadd_donation(p_member_idINT, p_amountDECIMAL);
FUNCTIONget_total_donations(p_member_idINT) RETURNDECIMAL;
ENDdonation_pkg;
```

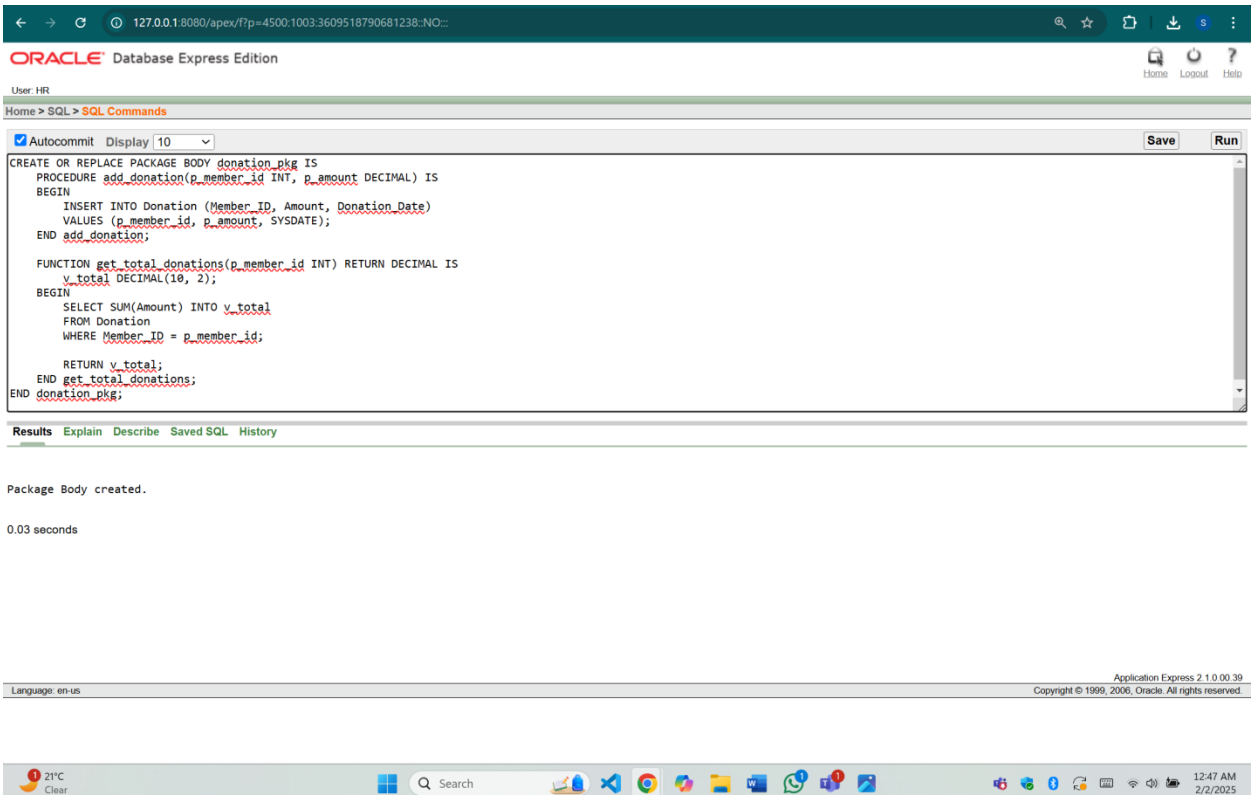


Query 2: Package body for donation operations

- **Question:** How does the package body implement the `add_donation` procedure and the `get_total_donations` function?

```
CREATEOR REPLACE PACKAGE BODY donation_pkgIS
PROCEDUREadd_donation(p_member_idINT, p_amountDECIMAL) IS
BEGIN
INSERTINTO Donation (Member_ID, Amount, Donation_Date)
VALUES (p_member_id, p_amount, SYSDATE);
ENDadd_donation;

FUNCTIONget_total_donations(p_member_idINT) RETURNDECIMALIS
v_totalDECIMAL(10, 2);
BEGIN
SELECTSUM(Amount) INTOv_total
```

```
FROM Donation
WHEREMember_ID=p_member_id;

RETURNv_total;
ENDget_total_donations;
ENDdonation_pkg;
```



Project Proposal: Welfare Club Management System

The proposed Welfare Club Management System is designed to efficiently manage the operations of a welfare club, focusing on event management, member administration, and financial tracking. It will provide two types of users Admin and Member. This system will simplify the process of organizing events, managing members, and tracking donations, ensuring smooth club operations and enhanced member engagement. By using technology and data analysis, the system aims to improve transparency, participation, and financial management within the club.

The system comprises several key components, including:

- **Event Management:** The Event Table stores information about club events, including event name, description, date, and the admin responsible for organizing the event. Each event is associated with a unique event ID, allowing for efficient event tracking and management.
- **Member Management:** The Member Table records details of club members, including member ID, name, email, address, phone number, and membership date. This information helps the admin manage member records and communicate effectively with members.
- **Participation Management:** The Participation Table tracks member participation in events. Each participation record includes a participation ID, event ID, and member ID, ensuring accurate tracking of member involvement in club activities.
- **Donation Management:** The Donation Table records donations made by members or external donors. Each donation record includes a donation ID, member ID, donor name, amount, and donation date. This data is used for financial tracking and reporting.
- **Admin Management:** The Admin Table manages admin information, including admin ID, name, email, and password. Admins have the authority to create and manage events, add or remove members, and oversee financial records.
- **Financial Tracking:** The system provides a comprehensive financial tracking mechanism, allowing admins to monitor donations and expenses related to events. This ensures transparency and accountability in club finances.
- **User Authentication and Access Control:** The system includes a secure login mechanism. Members can access event details, while admins have higher privileges to manage system functionalities and oversee operations.

Scope The system includes two types of users: Admin and Member

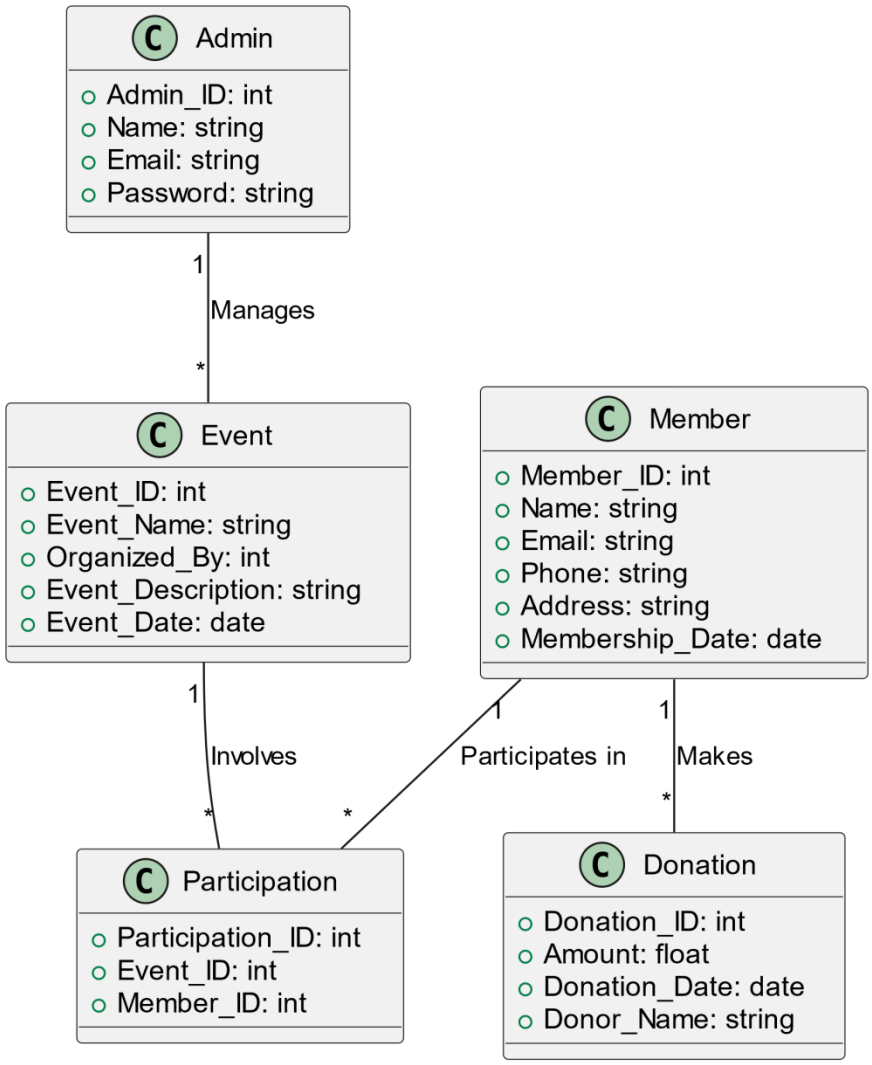
- **Admin**:
  - Create and manage events.
  - Manage member records.
  - Add new members.
  - Monitor financial transactions, including donations and expenses.
- **Member**:
  - View upcoming events.
  - Register and participate in events.
  - Make donations for events.

Technologies Used:

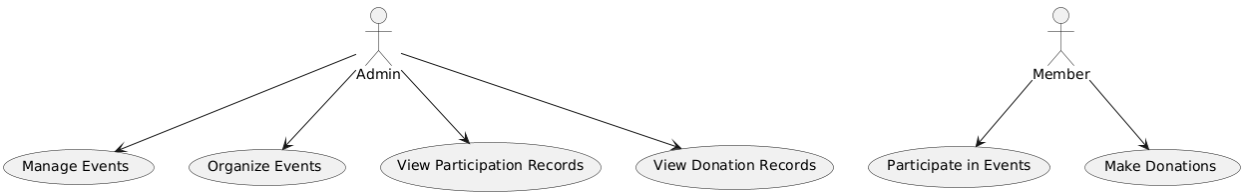
- **Frontend:** HTML, CSS
- **Backend:** Php
- **Database:** Oracle Database 10g

By implementing this system, welfare clubs can enhance their operational efficiency, increase member engagement, and maintain financial integrity. The structured approach ensures seamless event coordination, effective member management, and a transparent financial system. Ultimately, the Welfare Club Management System will contribute to the smooth functioning of welfare clubs, fostering better organization and community involvement.

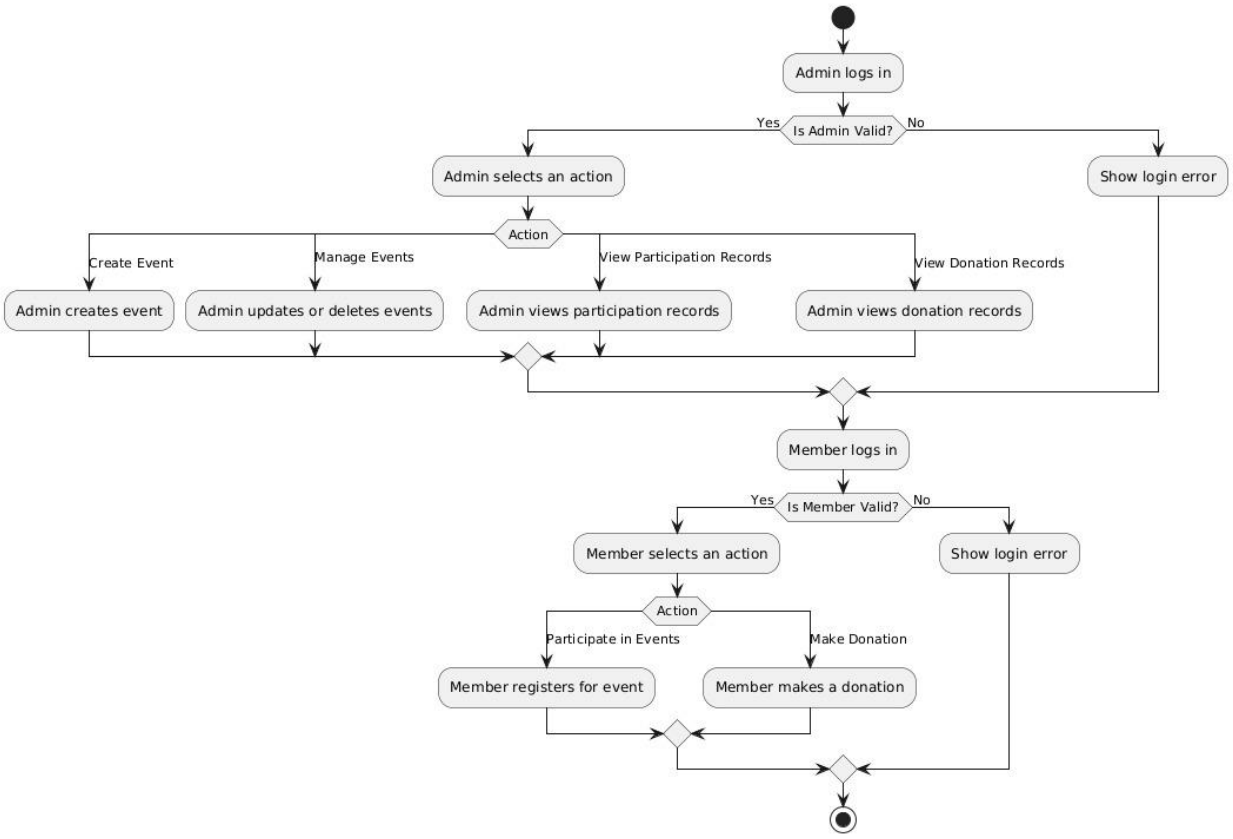
Diagram



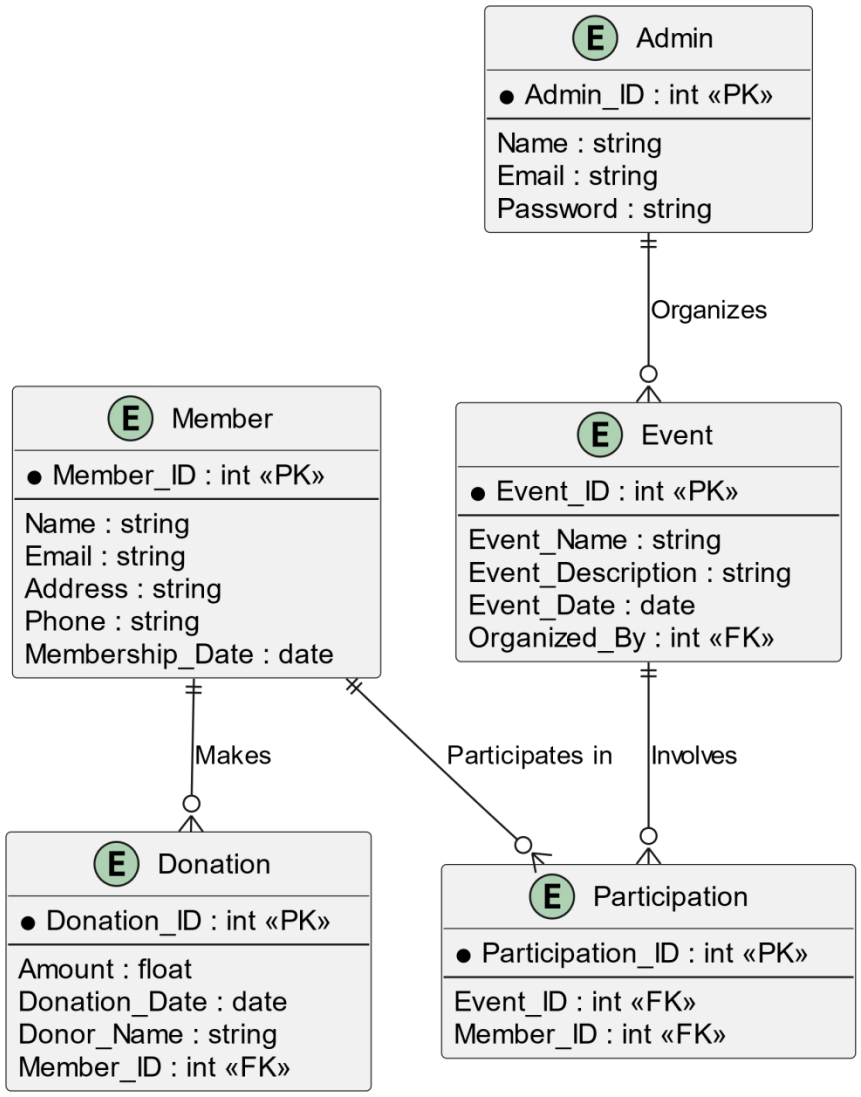
Class Diagram



Use Case Diagram

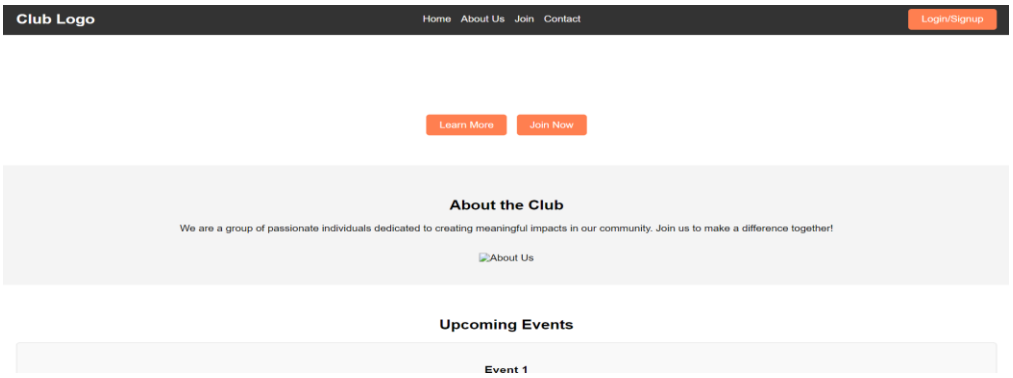


Activity Diagram



Schema Diagram

User Interface



Login

Login as:

User

Username:

Password:

Login

Welcome to the Welfare Club Member Dashboard

Profile | Upcoming Events | Donations | Volunteer Activities | Logout

Here you can manage your profile, view upcoming events, contribute donations, and participate in volunteer activities.

My Profile

Name: Mohammad rafi

Email: rafi@example.com

Phone: +880 123 456 7890

Membership Details

Role: Club Member

Membership ID: BD20231234

Join Date: March 5, 2023

Dues: Paid

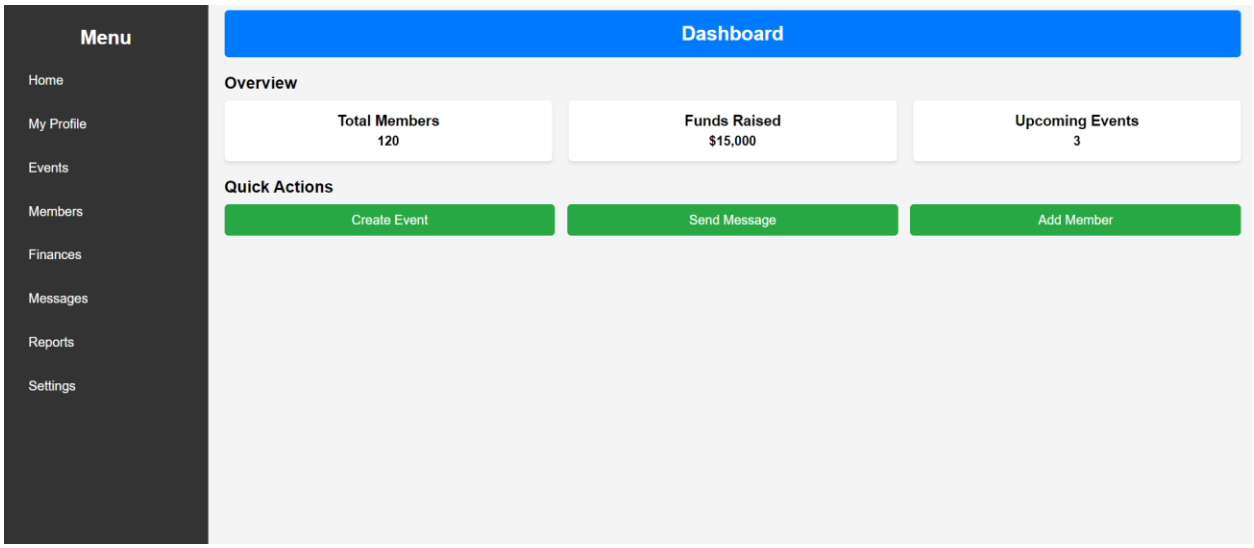
Edit Profile

Activity Log

Joined the "Tree Plantation Drive" on Dec 1, 2023

Attended "Leadership Training Workshop"

Messaged about member registration updates



Upcoming Events

Fundraising Workshop

Date: December 20, 2023

Time: 10:00 AM - 2:00 PM

View Details

Annual Club Meeting

Date: January 15, 2024

Time: 5:00 PM - 8:00 PM

View Details

Tree Plantation Drive

Date: December 25, 2023

Time: 9:00 AM - 12:00 PM

View Details

Create New Event

Event Title:

Enter event title

Date:

mm/dd/yyyy

Time:

Membership Management

Member Directory

Search: Search by name or role Status: Active Role: Member

Name: Rafi Patwari
Email: rafip@example.com
Role: Admin
Status: Active

Name: Mushi Khan
Email: mushi@example.com
Role: Member
Status: Inactive

Add New Member

Name:

Full Name

Email:

Email

Phone:

Phone Number

Role:

Member

Start Date:

mm/dd/yyyy

Add Member

Financial Management

Overview

Total Funds
$50,000

Recent Transactions
- Payment Received: $5000
- Expense Paid: $1200

Outstanding Dues
$2000

Transaction List

Type: All Date Range: mm/dd/yyyy mm/dd/yyyy

Date	Type	Category	Description	Amount
2024-12-01	Income	Donation	Donation received from donor X	$5000
2024-12-05	Expense	Event	Event cost for venue booking	$1200

Add New Transaction

Amount:

Amount

Type:

Income

Category:

Category

Description:

Transaction Description

Add Transaction

Budget Planning

Event or Cause:

Event Name

Budget Amount:

Messaging System

Notifications

New message from Hashmi Khan

New comment on your post

Inbox

Name	Subject	Date
Hashmi Khan	Meeting Tomorrow	2024-12-12
Will Smith	Project Update	2024-12-11

Message Thread

Hashmi Khan
2024-12-12

Hello, we have a meeting tomorrow. Please confirm your availability.

You
2024-12-12

Sure, I am available tomorrow. What time is the meeting?

Compose Message

To:

Hashmi Khan

Subject:

Subject

Message Body:

Write your message here...

Send Message

Reports

Predefined Report Templates

Membership Growth

A report showing the growth of members over time.

Generate Report

Event Attendance

A report showing attendance statistics for events.

Generate Report

Financial Summary

A report summarizing the financial status.

Generate Report

Custom Report Builder

Select Report Type:

Event Attendance

Date Range:

mm/dd/yyyy

mm/dd/yyyy

Filters:

Add filter criteria

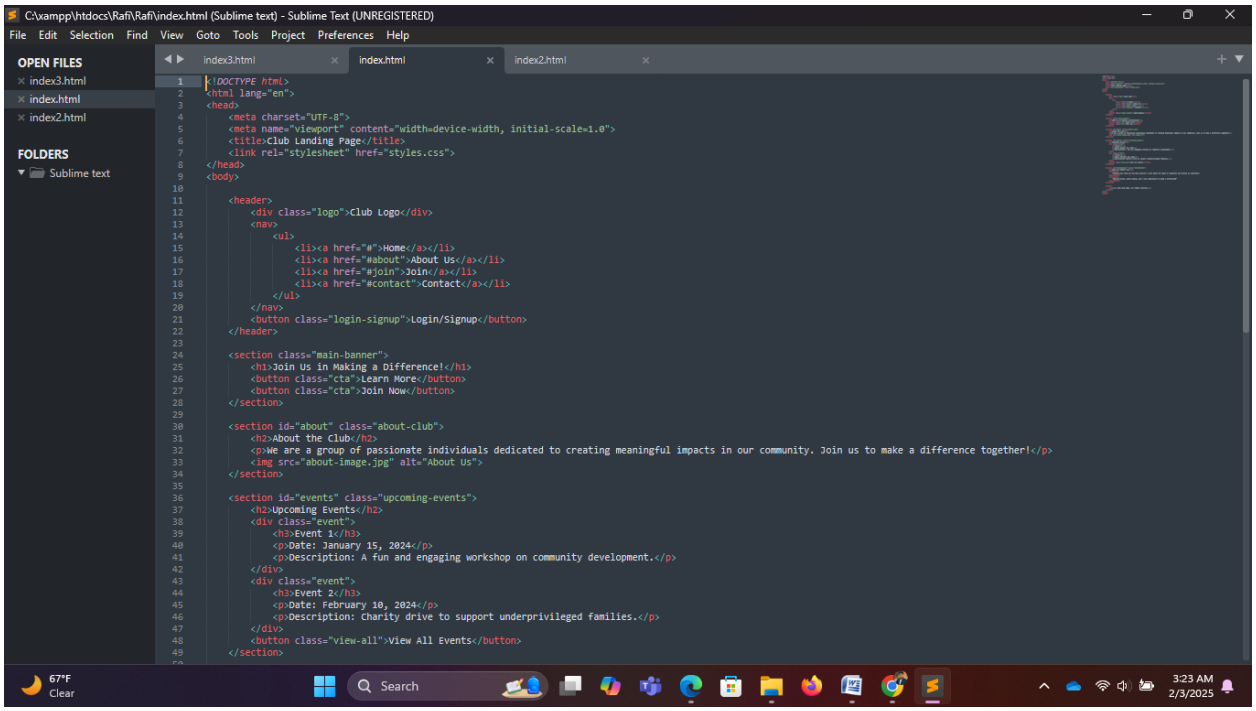
Select Data Fields:

Members
Events
Transactions

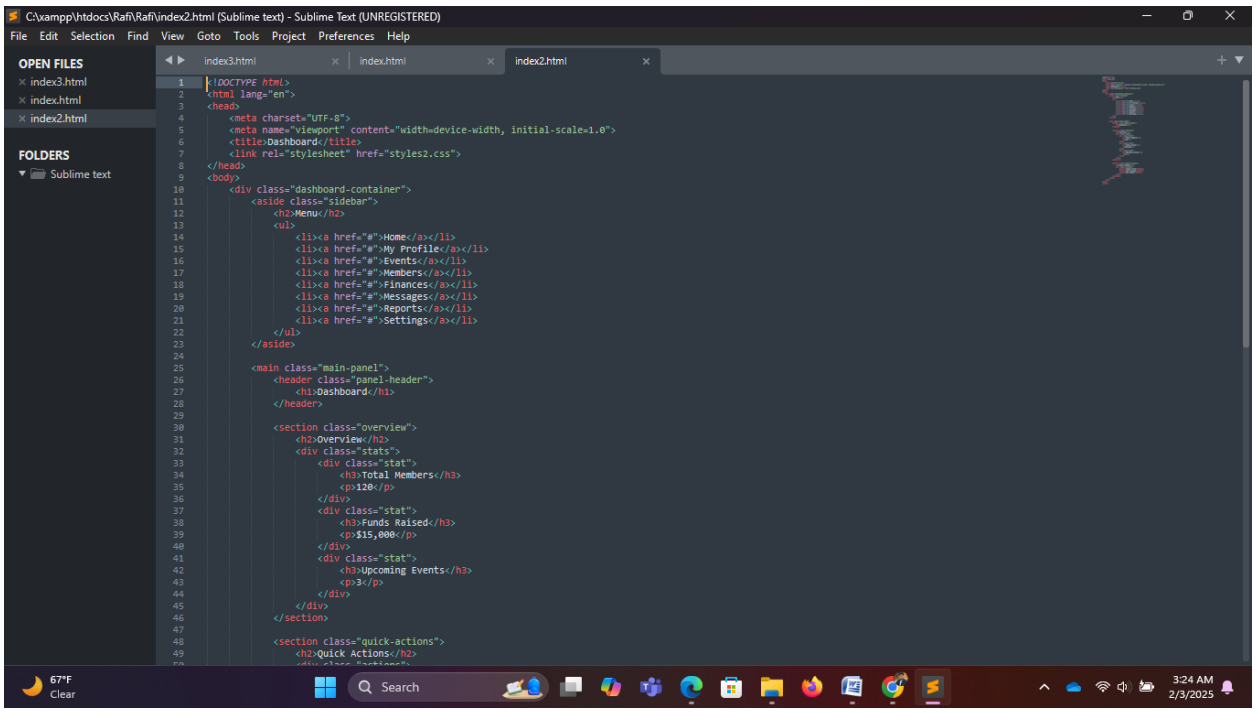
Export Options:

PDF

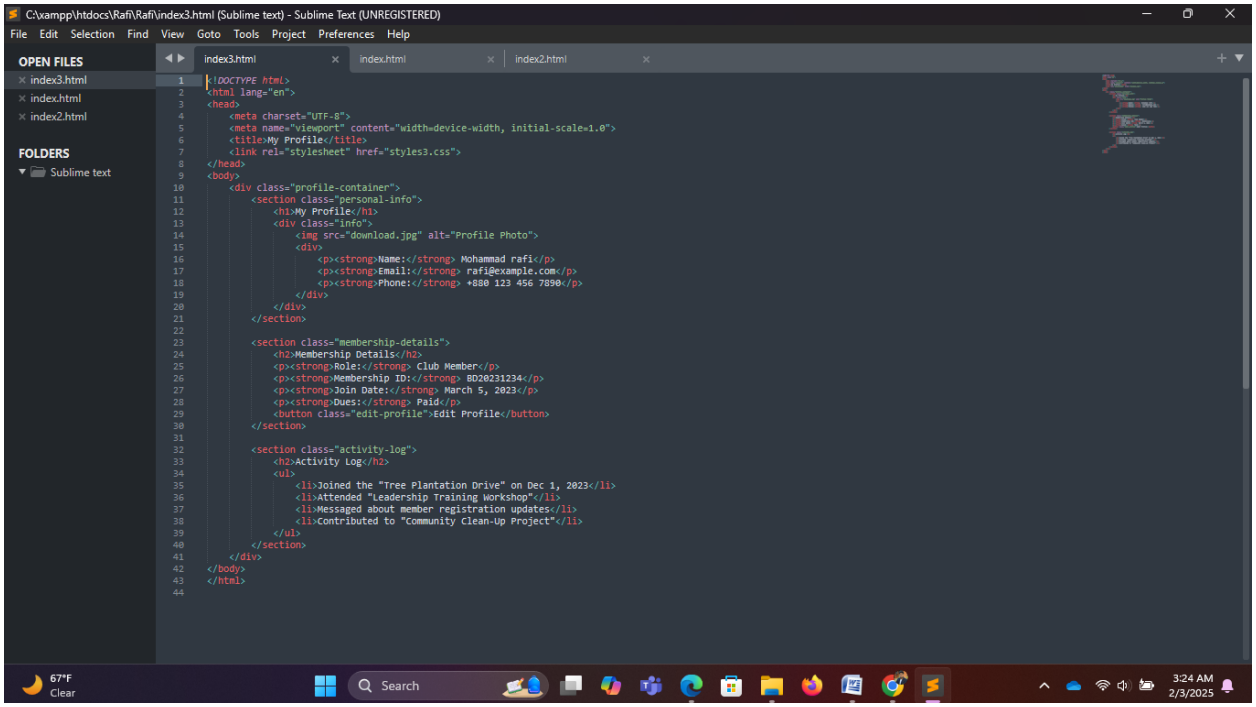
Generate Custom Report



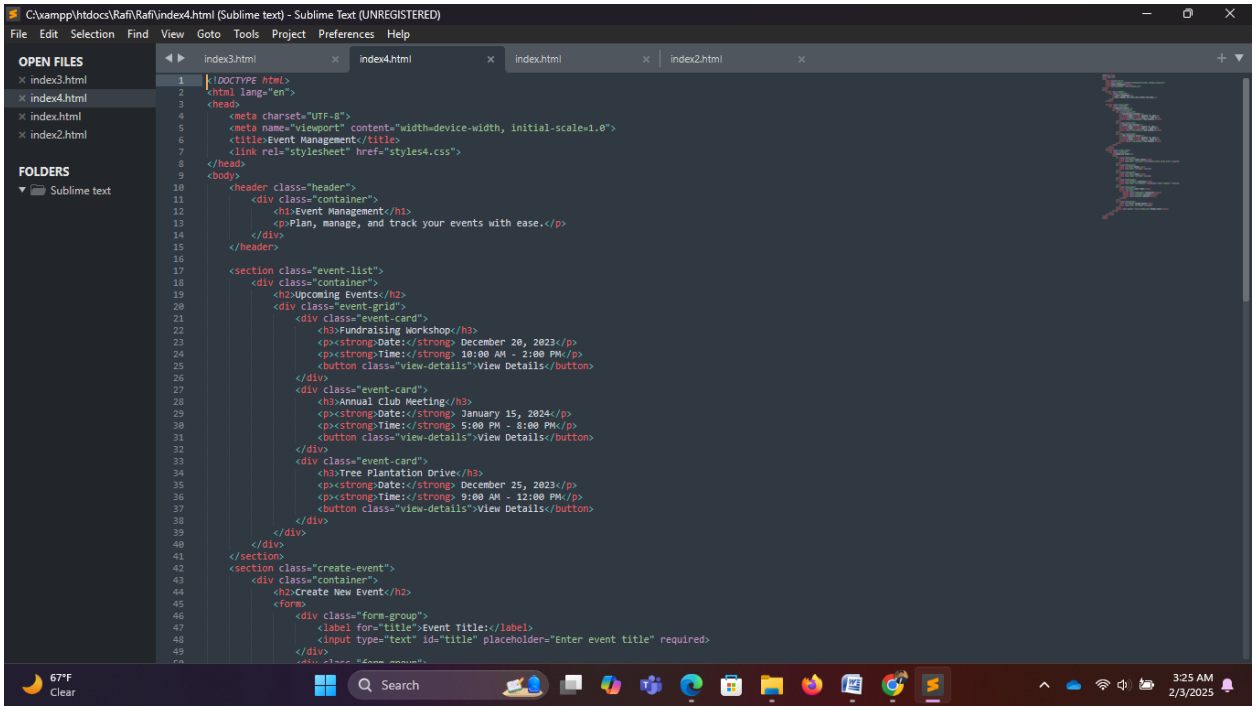
Code: Home page



Code: Admin Dashbord



Code : Member profile



Code : Event Management

Database Connection

Database Connection and Technical Issue

We implemented the Welfare Club Management System with a fully functional user interface (UI), database structure, and user verification system. We developed all the functionalities using php, and we used Oracle 10g as our database. However, we faced an unexpected technical issue while attempting to establish a connection between our php application and Oracle 10g database.

Database Connection work we done

To integrate Oracle 10g with our system, we followed these steps:

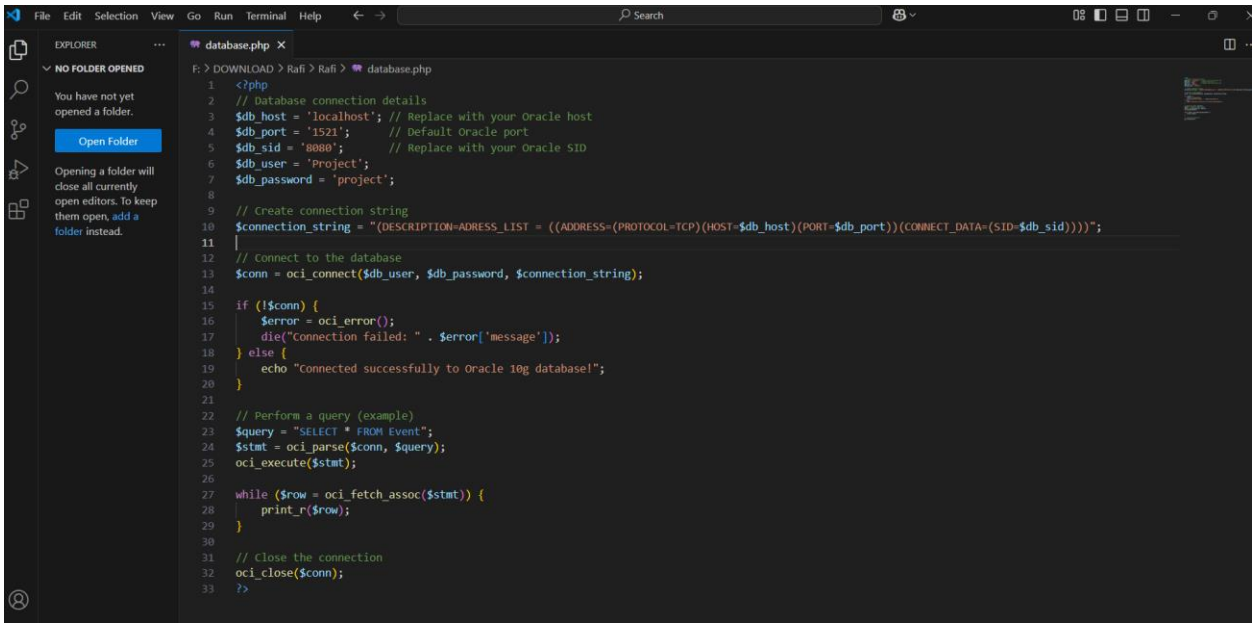
1. Installed Oracle 10g and configured the database.
2. Used php to establish a connection.
3. Attempted to configure connection settings using Oracle Client, SQL*Plus, or ODBC/JDBC.
4. Maybe We have problem with the credentials, hostname, port, and service name in the connection string.
5. For Debugging we find solution from google, youtube and chatgpt but we failed.

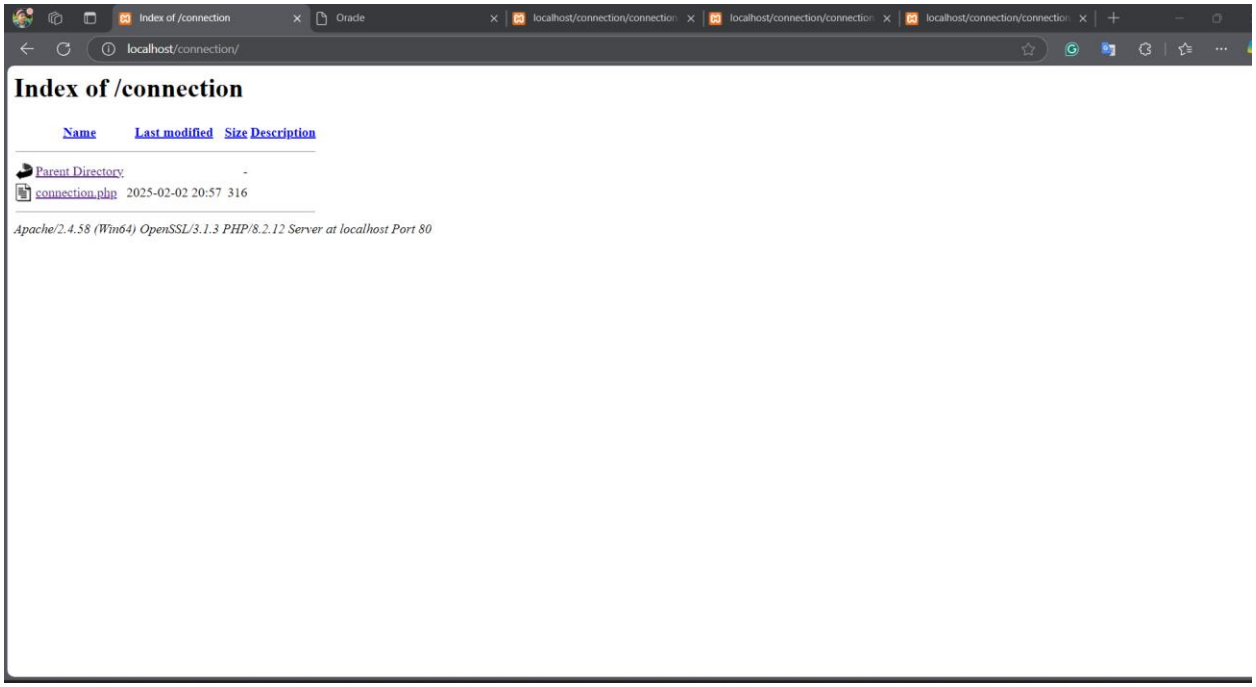
3. Challenges Faced in Database Connection

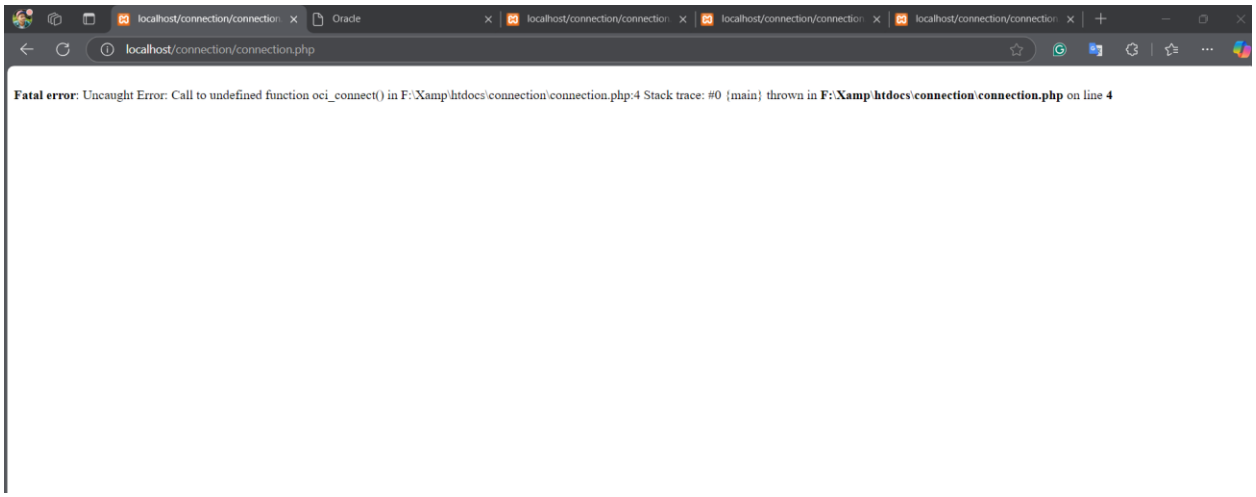
Challenges Faced

Despite following the correct steps, we encountered unexpected technical issues that prevented the system from connecting to Oracle 10g. Some of the difficulties included:

- Configuration errors in connecting JavaScript with Oracle 10g.


File Edit Selection View Go Run Terminal Help
Search
EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
database.php
F: > DOWNLOAD > Rafi > Rafi > database.php
<?php
// Database connection details
$db_host = 'localhost'; // Replace with your Oracle host
$db_port = '1521';   // Default Oracle port
$db_sid = '8080';   // Replace with your Oracle SID
$db_user = 'Project';
$db_password = 'project';
// Create connection string
$connection_string = "(DESCRIPTION=ADRESS_LIST = ((ADDRESS=(PROTOCOL=TCP)(HOST=$db_host)(PORT=$db_port))(CONNECT_DATA=(SID=$db_sid))))";
// Connect to the database
$conn = oci_connect($db_user, $db_password, $connection_string);
if (!$conn) {
$error = oci_error();
die("Connection failed: " . $error['message']);
} else {
echo "Connected successfully to Oracle 10g database!";
}
// Perform a query (example)
$query = "SELECT * FROM Event";
$stmt = oci_parse($conn, $query);
oci_execute($stmt);
while ($row = oci_fetch_assoc($stmt)) {
print_r($row);
}
// Close the connection
oci_close($conn);
?>


Index of /connection
Oracle
localhost/connection/connection
localhost/connection/connection
localhost/connection/connection
localhost/connection/
Index of /connection
Name Last modified Size Description
Parent Directory -
connection.php 2025-02-02 20:57 316
Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12 Server at localhost Port 80



4. Conclusion and Future Work

In the conclusion, acknowledge the issue and state that resolving it will be a priority in the future.

Our project successfully implements the core functionalities of the Welfare Club Management System, with a working user interface and backend. Although we faced technical difficulties in establishing the Oracle 10g connection, we have documented our attempts and challenges. In the future, further debugging and testing will be done to resolve this issue and ensure a fully functional database integration.

Query Writing Final Term:

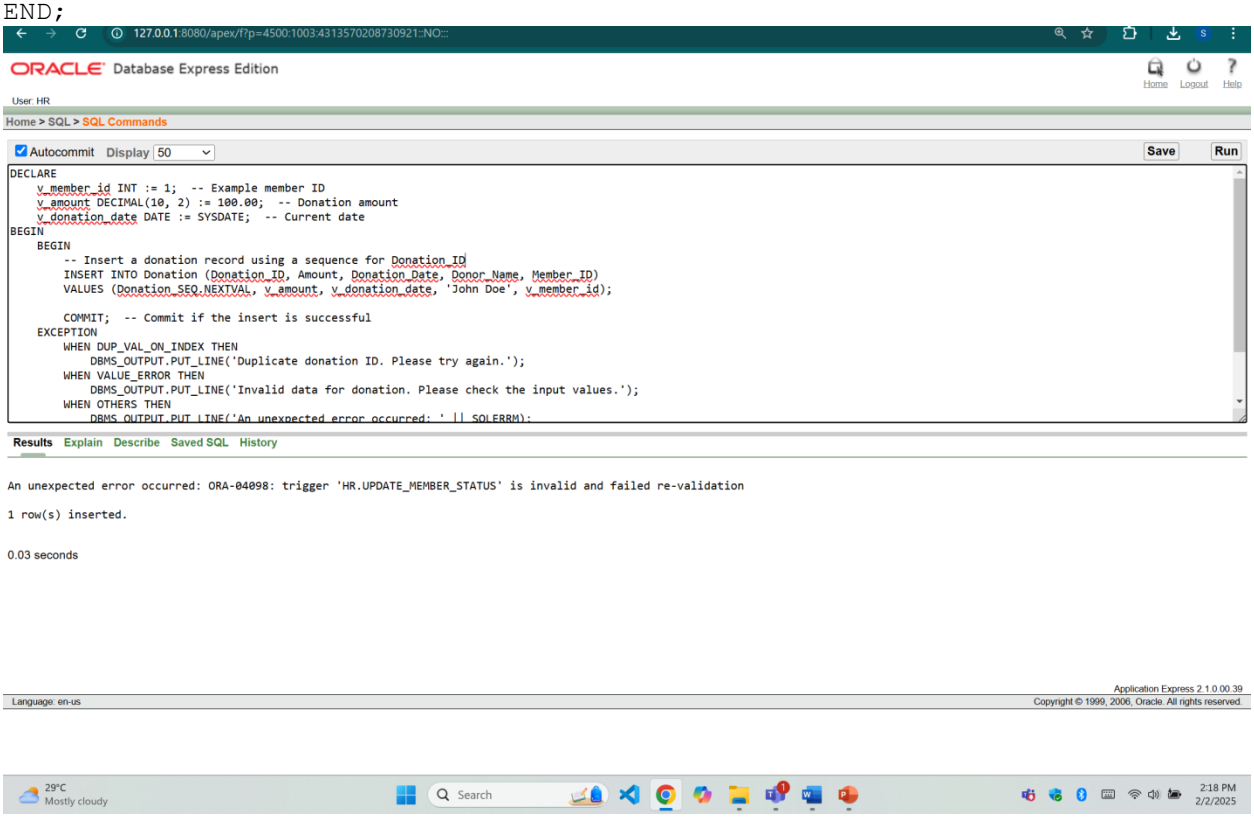
1. Exception Handling

Query 1: Exception Handling for Inserting a Donation

Question: **How does the exception handling mechanism in the `insert` operation manage possible errors such as duplicate values or invalid data?**

```
DECLARE
v_member_idINT :=1;
v_amountDECIMAL(10, 2) :=100.00;
v_donation_dateDATE := SYSDATE;
BEGIN
BEGIN
INSERTINTO Donation (Donation_ID, Amount, Donation_Date, Donor_Name,
Member_ID)
VALUES (Donation_SEQ.NEXTVAL, v_amount, v_donation_date, 'John Doe',
v_member_id);
COMMIT;
    EXCEPTION
```

```
WHEN DUP_VAL_ON_INDEX THEN
            DBMS_OUTPUT.PUT_LINE('Duplicate donation ID. Please try again.');
WHEN VALUE_ERROR THEN
            DBMS_OUTPUT.PUT_LINE('Invalid data for donation. Please check the
input values.');
WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: '|| SQLERRM);
END;
END;
```

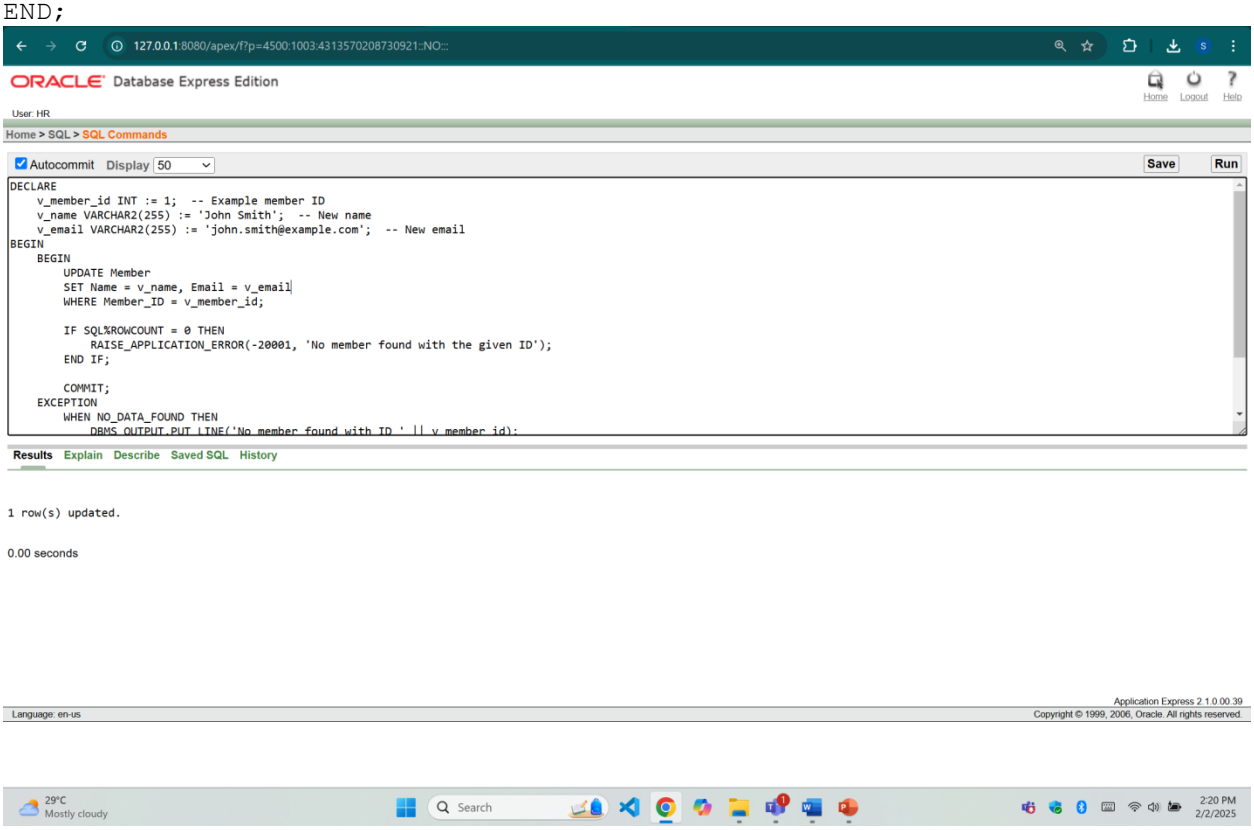


Query 2: Exception Handling for Updating a Member's Info

- **Question:** How does the `UPDATE` operation handle errors such as the absence of the member ID, and what does it log?

```
DECLARE
v_member_idINT :=1;  -- Example member ID
v_name VARCHAR2(255) :='John Smith';  -- New name
v_email VARCHAR2(255) :='john.smith@example.com';  -- New email
BEGIN
BEGIN
UPDATEMember
SET Name =v_name, Email =v_email
WHEREMember_ID=v_member_id;

        IF SQL%ROWCOUNT =0THEN
```

```
            RAISE_APPLICATION_ERROR(-20001, 'No member found with the given
ID');
END IF;

COMMIT;
    EXCEPTION
WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('No member found with ID '||v_member_id);
WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('An error occurred: '|| SQLERRM);
END;
END;
```



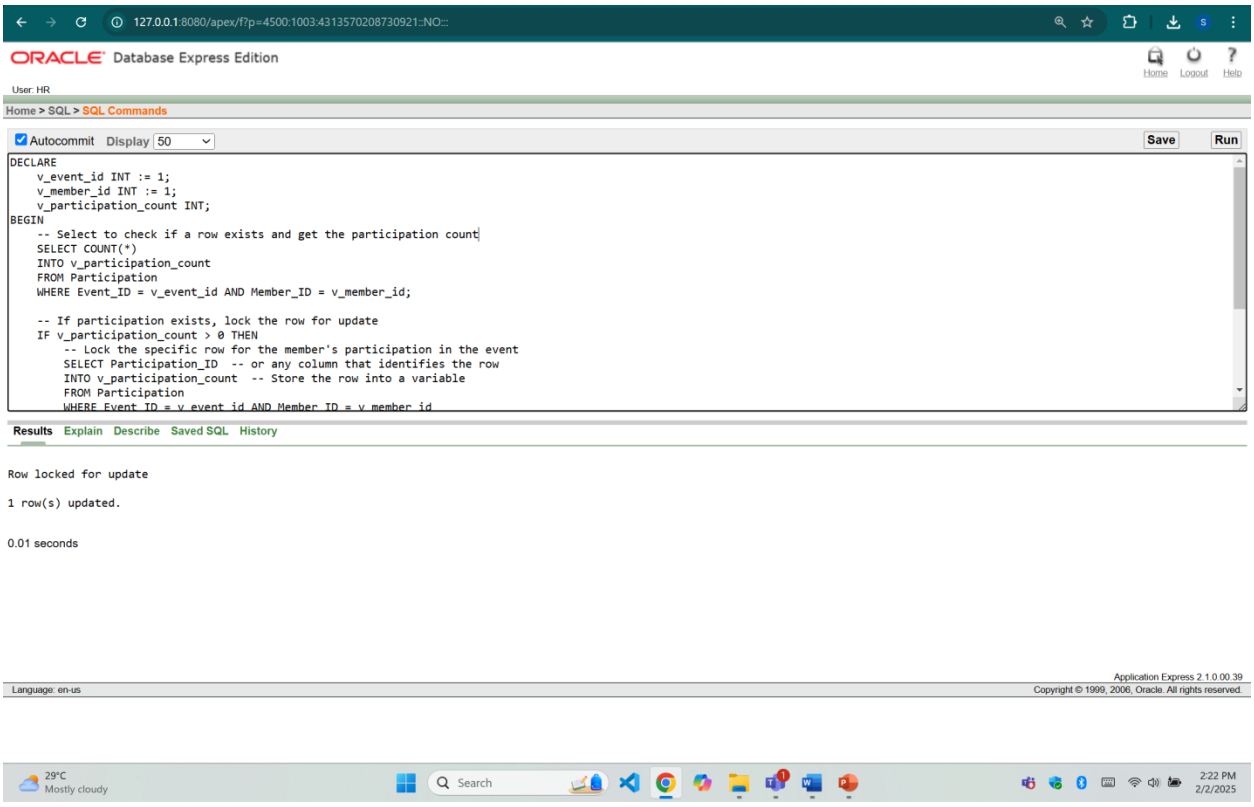
2. Implicit Locking

Query 1: Using **`SELECT FOR UPDATE`** to Implicitly Lock Rows

Question: **How does the `SELECT FOR UPDATE` statement implicitly lock the rows in the `Participation` table for further updates?**

```
DECLARE
v_event_id INT := 1;
v_member_id INT := 1;
v_participation_count INT;
BEGIN
    -- Select to check if a row exists and get the participation count
    SELECT COUNT(*)
```

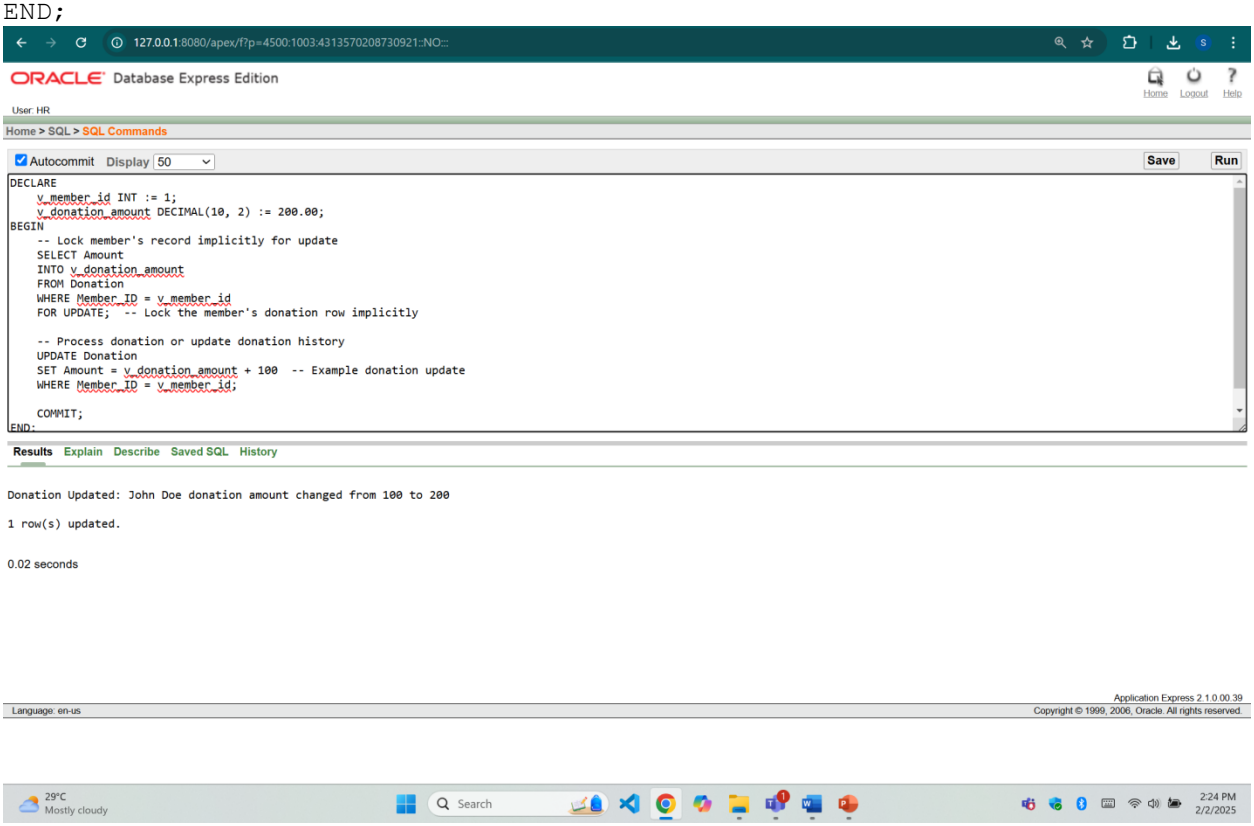
```
    INTO v_participation_count
    FROM Participation
    WHERE Event_ID = v_event_id AND Member_ID = v_member_id;

    -- If participation exists, lock the row for update
    IF v_participation_count> 0 THEN
        -- Lock the specific row for the member's participation in the event
        SELECT Participation_ID  -- or any column that identifies the row
        INTO v_participation_count  -- Store the row into a variable
        FROM Participation
        WHERE Event_ID = v_event_id AND Member_ID = v_member_id
        FOR UPDATE;  -- Lock the row for update

        -- Now, proceed with further operations (e.g., update participation,
etc.)
        DBMS_OUTPUT.PUT_LINE('Row locked for update');
    ELSE
        DBMS_OUTPUT.PUT_LINE('No participation found for the given event and
member.');
    END IF;
END;
```



Query 2: Implicit Locking During Donation Processing

- **Question:** How does the `SELECT FOR UPDATE` in the donation processing scenario lock a row for further updates on the donation table?

```
DECLARE
v_member_idINT :=1;
v_donation_amountDECIMAL(10, 2) :=200.00;
BEGIN
-- Lock member's record implicitly for update
SELECT Amount
INTOv_donation_amount
FROM Donation
WHEREMember_ID=v_member_id
FORUPDATE;  -- Lock the member's donation row implicitly

-- Process donation or update donation history
UPDATE Donation
SET Amount =v_donation_amount+100-- Example donation update
WHEREMember_ID=v_member_id;

COMMIT;
END;
```

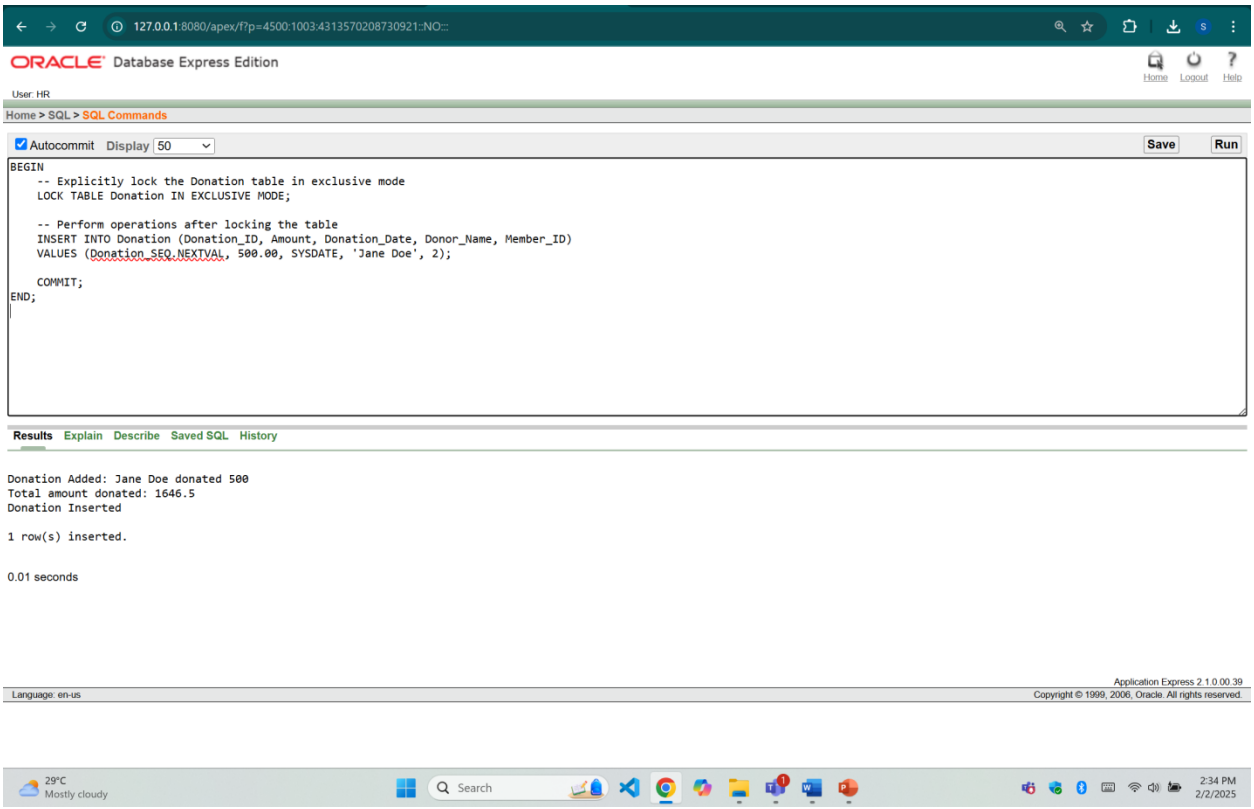


3. Explicit Locking

Query 1: Explicit Locking Using **`LOCK TABLE`**

Question: **How does the `LOCK TABLE` statement lock the `Donation` table for exclusive access during the insert operation?**

```
BEGIN
-- Explicitly lock the Donation table in exclusive mode
    LOCK TABLE Donation IN EXCLUSIVE MODE;

-- Perform operations after locking the table
INSERTINTO Donation (Donation_ID, Amount, Donation_Date, Donor_Name,
Member_ID)
VALUES (Donation_SEQ.NEXTVAL, 500.00, SYSDATE, 'Jane Doe', 2);

COMMIT;
END;
```



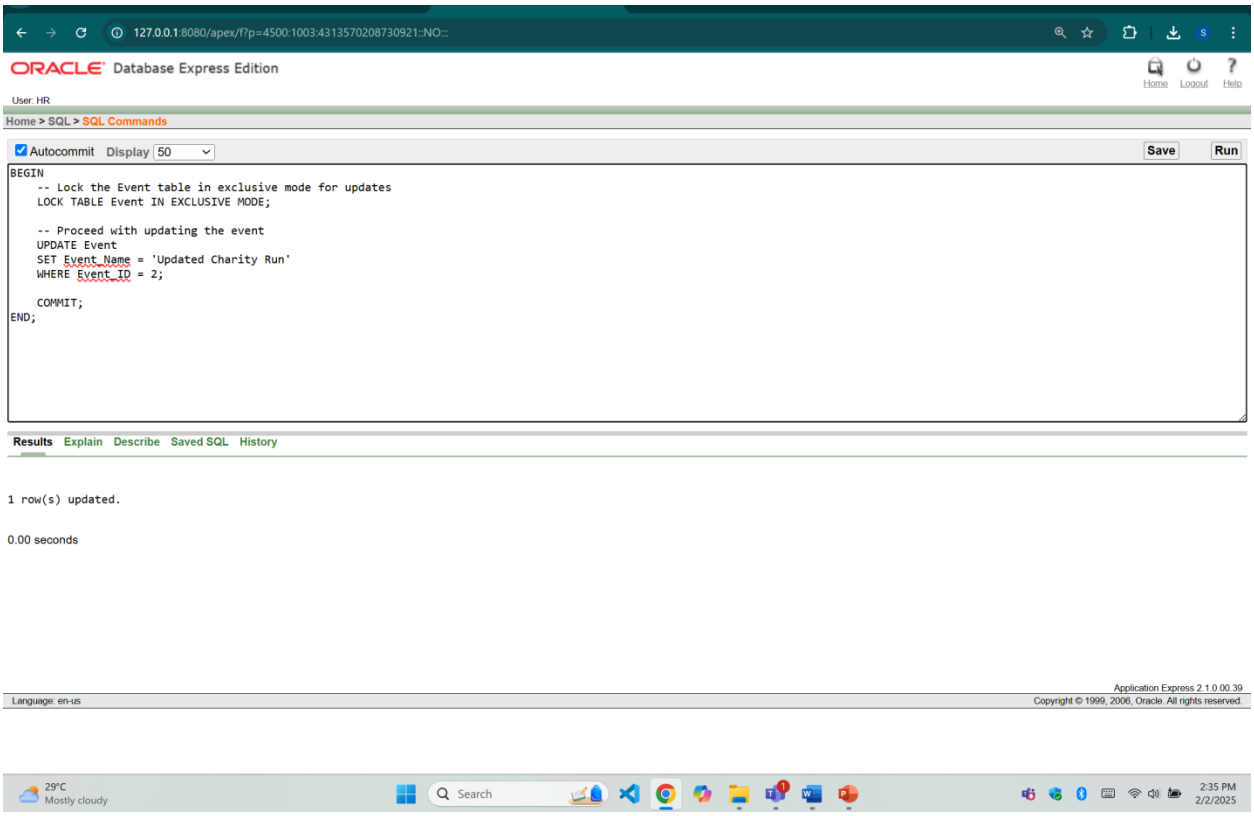
.

Query 2: Explicit Locking During Event Participation Update

- **Question:** How does the `LOCK TABLE` statement lock the `Event` table for exclusive access during an event update?

```
BEGIN
-- Lock the Event table in exclusive mode for updates
    LOCK TABLE Event IN EXCLUSIVE MODE;
```

```
-- Proceed with updating the event
UPDATE Event
SETEvent_Name='Updated Charity Run'
WHEREEvent_ID=2;

COMMIT;
END;
```



Relational Algebra:

1. Find all members who have donated more than 500 Taka. (Selection)

Ans: σ donation_amount > 500 (Donations)

2. Retrieve only the event name and event date from the Events table. (Projection)

Ans: ∏(event_name, event_date) (Events)

3. Find the total amount donated by each member. (Aggregation - Sum Function)

Ans: γ member_id, SUM(donation_amount) (Donations)

4. Find all events that have received donations and list their event IDs. (Intersection)

Ans: ∏(event_id) (Events) ∩ ∏(event_id) (Donations)

5. Rename the column "event_date" in the Events table to "date_of_event". (Rename)

Ans: ρ{Events(event_id, event_name, date_of_event, location)} (Events)

Conclusion: The Welfare Club Management System improves efficiency by automating event management, member coordination, and financial tracking. It ensures secure and transparent operations, reducing manual efforts and errors. By providing a structured and user-friendly platform, the system enhances overall club management and member engagement. However, there are still scopes of improvement through some future works.

Proposed Future Work:

- **Advanced Features:** Incorporate real-time event tracking, data analytics, and better reporting tools to optimize usability.