



**American International University-Bangladesh**

### **Final Term Project Documentation**

**Project Name: welfare club Management System**

**Course Name: Advance Database Management System**

### **Section: B**

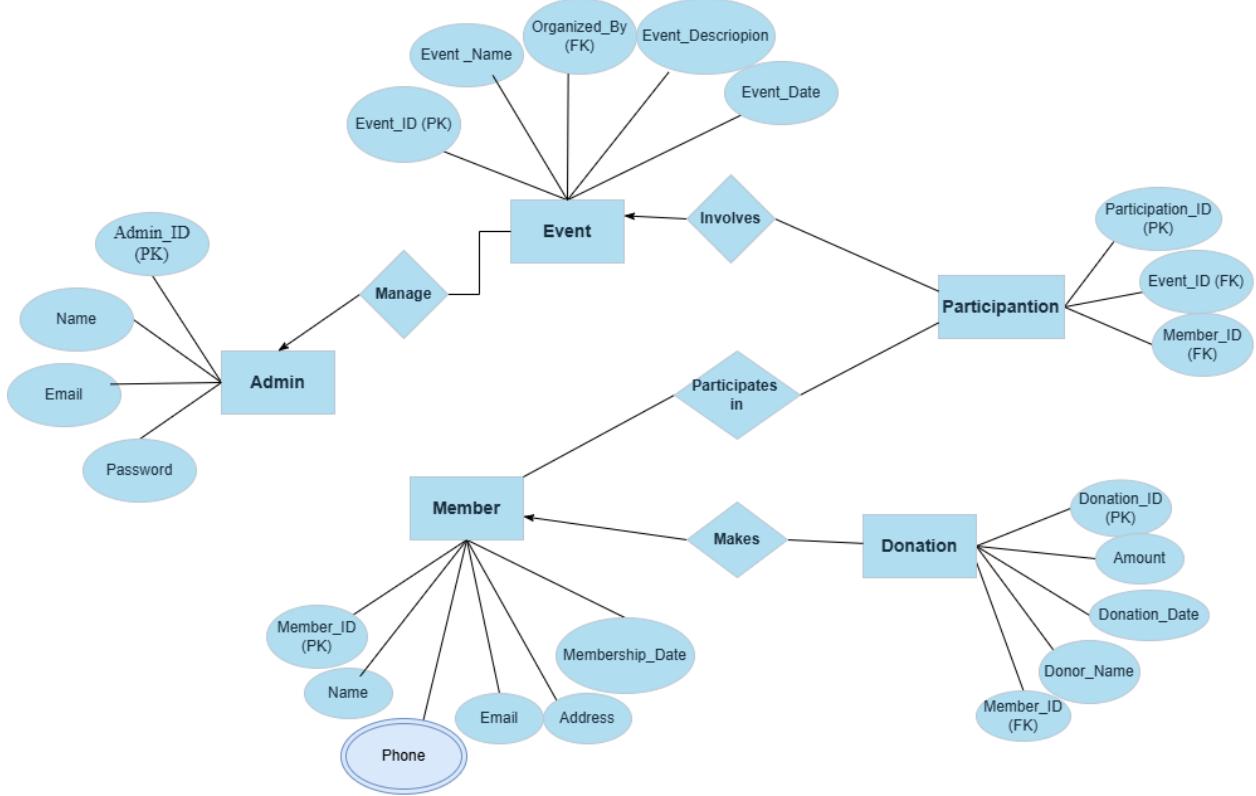
Serial No	Name	ID	Contributed Percentage	Topic Names
01	MD. ABDULLAH AL MAHMUD NAIEM	21-45255-2	25%	Project planning, project proposal, Relational Algebra, Report writing
02	SUMAIYA TAB	21-45928-3	25%	PI/SQL mid, Final
03	MST.JANNATUL FERDOUS	21-45440-3	25%	Class Diagram, Use Case Diagram, Activity Diagram, 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');
```

User: HR

Home > SQL > **SQL Commands**

Autocommit

```
SELECT * FROM Admin;
```

**Results** [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

ADMIN_ID	NAME	EMAIL	PASSWORD
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

5 rows returned in 0.02 seconds [CSV Export](#)

Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

```
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);
```

127.0.0.1:8080/apex/?p=4500:1003:3609518790681238:NO::

**ORACLE® Database Express Edition**

User HR

Home > SQL > SQL Commands

Autocommit Display | 10 |

```
SELECT*FROM Event;
```

**Results Explain Describe Saved SQL History**

EVENT_ID	EVENT_NAME	EVENT_DESCRIPTION	EVENT_DATE	ORGANIZED_BY
1	Fundraiser Gala	An evening of fundraising for the community.	15-MAY-25	1
2	Charity Run	A 5K run to raise awareness for cancer research.	10-JUN-25	2
3	Concert for Hope	A musical event for mental health awareness.	20-JUL-25	3
4	Food Drive	Collecting food for local food banks.	05-AUG-25	4
5	Art Exhibition	An exhibition to raise funds for local artists.	01-SEP-25	5

5 rows returned in 0.00 seconds [CSV Export](#)

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.



```
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'));
```

ORACLE Database Express Edition

User: HR

Home > SQL > SQL Commands

Autocommit Display: 10 |

```
SELECT * FROM Member ;
```

**Results Explain Describe Saved SQL History**

MEMBER_ID	NAME	EMAIL	ADDRESS	PHONE	MEMBERSHIP_DATE
1	Emily Adams	emily.adams@example.com	123 Elm St, Springfield, IL	555-1234	25-NOV-23
2	James Carter	james.carter@example.com	456 Oak St, Rivertown, OH	555-5678	15-JAN-24
3	Sophia Williams	sophia.williams@example.com	789 Pine St, Lakeside, FL	555-8765	10-MAR-24
4	Daniel Brown	daniel.brown@example.com	101 Maple St, Hilltop, CA	555-2345	01-JUL-23
5	Olivia Green	olivia.green@example.com	202 Birch St, Mountainview, TX	555-3456	20-MAY-24

5 rows returned in 0.00 seconds [CSV Export](#)

Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.



```
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);
```

ORACLE Database Express Edition

User HR

Home > SQL > SQL Commands

Autocommit Display 10

```
SELECT*FROM Participation;
```

Results Explain Describe Saved SQL History

PARTICIPATION_ID	EVENT_ID	MEMBER_ID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

5 rows returned in 0.01 seconds [CSV Export](#)

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.



```
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);
```

The screenshot shows the Oracle Database Express Edition interface. At the top, it says "ORACLE Database Express Edition". Below that, "User: HR" and "Home > SQL > SQL Commands". The SQL command entered is "SELECT \* FROM Donation;". The results section shows a table with 5 rows:

DONATION_ID	AMOUNT	DONATION_DATE	DONOR_NAME	MEMBER_ID
1	100	10-JAN-25	John Doe	1
2	250.5	01-FEB-25	Alice Smith	2
3	500	05-MAR-25	Bob Johnson	3
4	75.25	20-APR-25	Charlie Brown	4
5	120.75	10-MAY-25	David Lee	5

Below the table, it says "5 rows returned in 0.00 seconds" and "CSV Export".

Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.



## 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;

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code entered is:

```
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;
```

The results section shows the output:

```
Admin Name: John Doe
Statement processed.
```

Execution details:

- Language: en-us
- Time: 6:55 PM
- Date: 2/1/2025
- Duration: 0.03 seconds

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## 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;
```

127.0.0.1:8080/apex/f?p=4500:1003:3609518790681238::NO::

ORACLE Database Express Edition

User HR

Home > SQL > SQL Commands

Autocommit Display 10

```
DECLARE
    v_event_name VARCHAR2(255); -- Declare variable to hold event name
BEGIN
    -- Use the variable to get the event name by Event_ID
    SELECT Event_Name INTO v_event_name FROM Event WHERE Event_ID = 3;
    -- Output the event name
    DBMS_OUTPUT.PUT_LINE('Event Name: ' || v_event_name);
END;
```

Results Explain Describe Saved SQL History

Event Name: Concert for Hope  
Statement processed.  
0.03 seconds

Language en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.



## 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');
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. A SQL query is entered in the command window:

```
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');
```

The results window displays the output:

EVENT_NAME	EVENT_DATE
Charity Run	10-JUN-25
Concert for Hope	20-JUL-25

2 rows returned in 0.02 seconds

CSV Export

Language: en-us Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.

## 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;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The URL is 127.0.0.1:8080/apex/f?p=4500:1003:609510:90881238::NO... . The interface includes a toolbar with Home, Logout, and Help buttons. The main area shows a SQL command window with the following code:

```
SELECT Donor_Name, Amount
FROM Donation
WHERE Amount > 100;
```

Below the code, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected, displaying a table with the following data:

DONOR_NAME	AMOUNT
Alice Smith	250.5
Bob Johnson	500
David Lee	120.75

Text at the bottom indicates "3 rows returned in 0.00 seconds" and a "CSV Export" link. The status bar at the bottom right shows "Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved." and the system tray shows a weather icon (22°C Clear), a search bar, and various application icons.

### 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;
```

ORACLE Database Express Edition

User HR

Home > SQL > SQL Commands

Autocommit Display 10

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

Results Explain Describe Saved SQL History

UPPER_NAME
EMILY ADAMS

1 rows returned in 0.02 seconds CSV Export

Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## 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;
```

ORACLE Database Express Edition

User: HR

Home > SQL > SQL Commands

Autocommit Display 10

```
SELECT
    SUBSTR(Event_Name, 1, 5) AS Short_Name,
    ROUND(SYSDATE - Event_Date) AS Days_Since_Event
FROM Event
WHERE Event_ID = 1;
```

Results Explain Describe Saved SQL History

SHORT_NAME	DAY_SINCE_EVENT
Fundr	-102

1 rows returned in 0.02 seconds

Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2000, Oracle. All rights reserved.

## 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;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The URL is 127.0.0.1:8080/apex/f?p=4500:1003:3609518790681238::NO:::. The user is HR. The SQL command entered is:

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

The results are displayed in a table:

PARTICIPATION_COUNT	MAX_EVENT_ID
1	1

1 rows returned in 0.00 seconds CSV Export

Language: en-us Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved.

## 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;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. A user named HR is connected. The SQL command entered is:

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

The results pane shows a single row with the value 250.5 under the column header AVG\_DONATION. The status bar at the bottom indicates "1 rows returned in 0.00 seconds".

## 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;
```



The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code entered is:

```

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;

```

The output shows the results of the loop:

```

Event: Fundraiser Gala
Event: Charity Run
Event: Concert for Hope
Event: Food Drive
Event: Art Exhibition

```

Below the results, it says "Statement processed." and "0.01 seconds". At the bottom, it shows the Application Express version and copyright information.

## 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;

```



ORACLE Database Express Edition

User: IIR  
Home > SQL > SQL Commands

```

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;

```

**Results** Explain Describe Saved SQL History

```

Member: Emily Adams - Event: Fundraiser Gala
Member: James Carter - Event: Charity Run
Member: Sophia Williams - Event: Concert for Hope
Member: Daniel Brown - Event: Food Drive
Member: Olivia Green - Event: Art Exhibition

Statement processed.

0.02 seconds

```

Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## 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;

```

User HR

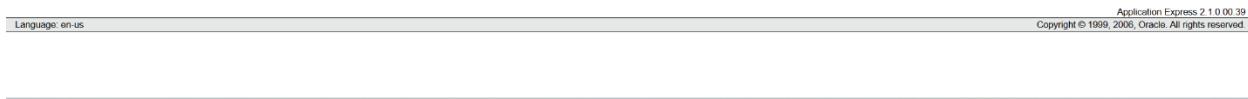
Home > SQL > SQL Commands

Autocommit Display 10

```
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;
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

It is a Fundraiser event!  
Statement processed.  
0.02 seconds



## 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;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The URL is 127.0.0.1:8080/apex/f?p=4500:1003:3609518790681238::NO:::. The page title is ORACLE Database Express Edition. The user is HR. The menu bar includes Home, Logout, Help, and a search icon. The main area shows a SQL command window with the following code:

```

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;

```

Below the code, there are buttons for Save and Run. The results section shows the output: "Small Donation." followed by "Statement processed." and "0.02 seconds". At the bottom right, it says Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.

## 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');

```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands editor, a query is run:

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

The results show one row:

NAME
Emily Adams

1 rows returned in 0.00 seconds

CSV Export

Language: en-us Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.

## 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;
```

User: HR

Home > SQL > SQL Commands

```
Autocommit Display 10
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;
```

Save Run

Results Explain Describe Saved SQL History

MEMBER_NAME	TOTAL_DONATIONS
Olivia Green	120.75
Sophia Williams	500
James Carter	250.5

3 rows returned in 0.00 seconds CSV Export

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.



## 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;
```

← → ⌂ 127.0.0.1:8080/apex/f?p=4500:1003:3609518790681238::NO::

ORACLE Database Express Edition

User HR

Home > SQL > SQL Commands

Autocommit Display | 10 |

```
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;
```

Results Explain Describe Saved SQL History

EVENT_NAME	PARTICIPATION_ID
Fundraiser Gala	1

1 rows returned in 0.00 seconds [CSV Export](#)

Language: en-us Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.

## 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;
```

The screenshot shows the Oracle Database Express Edition SQL Command window. The URL is 127.0.0.1:8080/apex/?p=4500.1003.3609518790681238::NO::. The window title is ORACLE Database Express Edition. The user is HR. The menu bar includes Home, Logout, Help. The toolbar has Save and Run buttons. The SQL command input field contains:

```
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;
```

The results pane shows a table with two columns: EVENT\_NAME and MEMBER\_NAME. The data is:

EVENT_NAME	MEMBER_NAME
Charity Run	James Carter

1 rows returned in 0.01 seconds. There is a CSV Export link.

At the bottom, the Application Express footer reads: Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.

## 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?

```
CREATE OR REPLACE FUNCTION get_participant_count (p_event_id INT)
RETURN INT IS
v_count INT;
BEGIN
SELECT COUNT(*) INTO v_count
FROM Participation
WHERE Event_ID = p_event_id;

RETURN v_count;
END;
```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a function named `get_participant_count` is being created. The code defines a parameter `p_event_id` of type `INT`, declares a local variable `v_count` of type `INT`, begins a block, selects the count of rows from the `Participation` table where `Event_ID` matches `p_event_id`, and returns the value of `v_count`. The function is saved and run, resulting in a message that says "Function created." and "0.00 seconds". The status bar at the bottom right indicates "Application Express 2.1.0.0.39" and "Copyright © 1999, 2008, Oracle. All rights reserved".

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

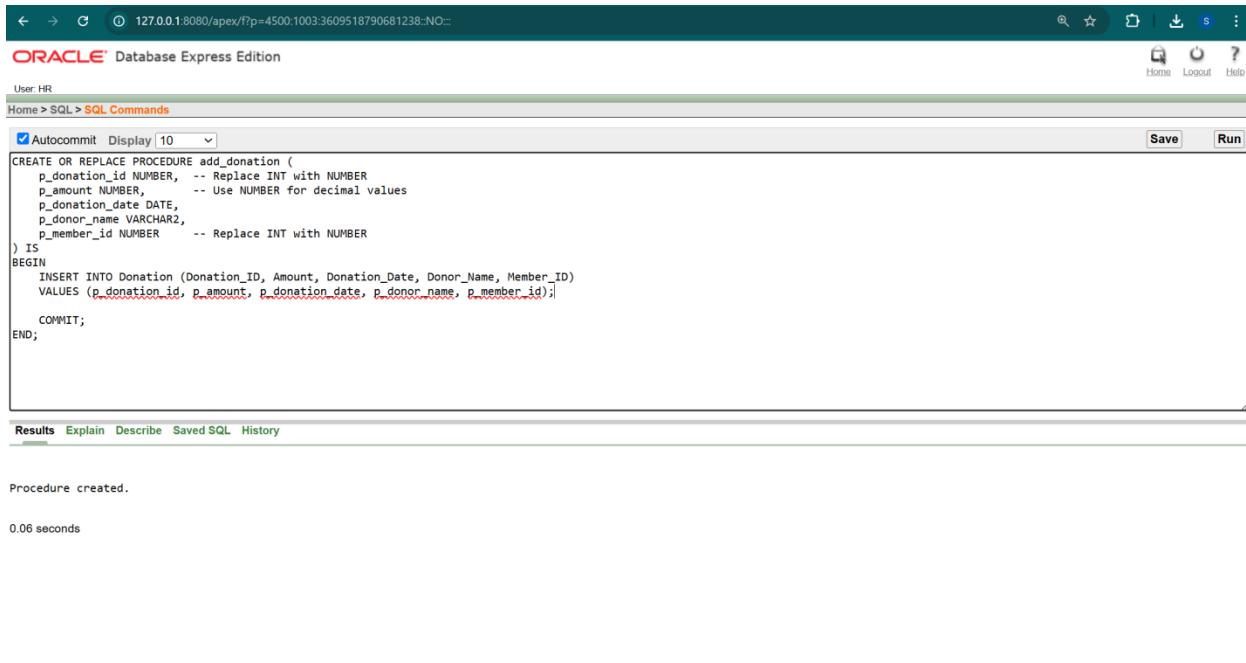
## 2. Stored Procedure

### 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;
```



The screenshot shows the Oracle Database Express Edition interface. The URL is 127.0.0.1:8080/apex/f?p=4500:1003:360951879061238::NO:::. The page title is "ORACLE Database Express Edition". The navigation bar includes Home, Logout, and Help. The main area is titled "SQL Commands" and contains the SQL code for the `add_donation` procedure. The code is identical to the one provided above. The "Run" button is visible at the top right of the code editor. Below the code, the message "Procedure created." is displayed, along with a timestamp of "0.06 seconds". At the bottom, there are links for Results, Explain, Describe, Saved SQL, and History. The status bar at the bottom right shows "Application Express 2.1.0.0.39", "Copyright © 1999, 2006, Oracle. All rights reserved.", and the date "2/1/2025".

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
UPDATERMember
SET Name =p_name,
        Email =p_email,
        Phone =p_phone
WHEREMember_ID=p_member_id;
```

127.0.0.1:8080/apex/f?p=4500:1003:3609518790681238:NO::

ORACLE Database Express Edition

User HR

Home > SQL > SQL Commands

Autocommit

```
CREATE OR REPLACE PROCEDURE update_member_details (
    p_member_id INT,
    p_name VARCHAR2,
    p_email VARCHAR2,
    p_phone VARCHAR2
) IS
BEGIN
    UPDATE Member
    SET Name = p_name,
        Email = p_email,
        Phone = p_phone
    WHERE Member_ID = p_member_id;
    COMMIT;
END;
```

Procedure created.

0.01 seconds

Language: en-us Application Express 2.1.0.00.39 Copyright © 1999, 2005 Oracle. All rights reserved.



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
INTO v_event.Event_ID, v_event.Event_Name, v_event.Event_Date
FROM Event
WHERE Event_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;

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code entered is the same as above, including the declaration of a record type and its usage in a PL/SQL block. The results pane displays the output of the DBMS\_OUTPUT.PUT\_LINE statements, which are the event details: Event ID: 1, Event Name: Fundraiser Gala, and Event Date: 15-MAY-25.

Language: en-us Application Express 2.1.0.0.39  
Copyright © 1999, 2006, Oracle. All rights reserved.



## 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)
INTO v_donation.Member_ID, v_donation.Donor_Name, v_donation.Total_Amount
FROM Donation d
WHERE d.Member_ID=1
GROUP BY d.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;

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code entered is a PL/SQL block that declares a record type, initializes a cursor, performs a SELECT query, and then uses DBMS\_OUTPUT.PUT\_LINE to print the member ID, donor name, and total amount. The results pane displays the output correctly.

```

User HR
Home > SQL > SQL Commands
Autocommit Display | 10 | Save | Run
DECLARE
  TYPE Donation_Record IS RECORD (
    Member_ID INT,
    Donor_Name VARCHAR2(255),
    Total_Amount DECIMAL(10,2)
  );
  v_donation Donation_Record;
BEGIN
  SELECT d.Member_ID, d.Donor_Name, SUM(d.Amount)
  INTO v_donation.Member_ID, v_donation.Donor_Name, v_donation.Total_Amount
  FROM Donation d
  WHERE d.Member_ID = 1
  GROUP BY d.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;

```

Results Explain Describe Saved SQL History

```

Member ID: 1
Donor Name: John Doe
Total Donation Amount: 100
Statement processed.

0.02 seconds

```



## 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;

```

The screenshot shows the Oracle Database Express Edition interface. The SQL Commands window contains the following PL/SQL code:

```

DECLARE
    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;

```

The results pane shows the output of the PL/SQL block:

```

Event ID: 1, Name: Fundraiser Gala, Date: 15-MAY-25
Event ID: 2, Name: Charity Run, Date: 10-JUN-25
Event ID: 3, Name: Concert for Hope, Date: 20-JUL-25
Event ID: 4, Name: Food Drive, Date: 05-AUG-25
Event ID: 5, Name: Art Exhibition, Date: 01-SEP-25
Statement processed.
0.02 seconds

```

The status bar at the bottom right indicates "Application Express 2.1.0.00.39" and "Copyright © 1999, 2006, Oracle. All rights reserved."

## 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
CURSOR donation_cursor IS

```

```

SELECT m.Name, SUM(d.Amount) AS Total_Donations
FROM Member m
JOIN Donation d ON m.Member_ID = d.Member_ID
GROUP BY m.Name;

v_member_record%ROWTYPE;
BEGIN
OPEN donation_cursor;

LOOP
FETCH donation_cursor INTO v_member_record;
EXIT WHEN donation_cursor%NOTFOUND;

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

```

```

CLOSE donation_cursor;
END;

```

User HR  
Home > SQL > SQL Commands

Autocommit Display: 10

```

DECLARE
CURSOR donation_cursor IS
    SELECT m.Name, SUM(d.Amount) AS Total_Donations
    FROM Member m
    JOIN Donation d ON m.Member_ID = d.Member_ID
    GROUP BY m.Name;

    v_member_record%ROWTYPE;
BEGIN
OPEN donation_cursor;

LOOP
    FETCH donation_cursor INTO v_member_record;
    EXIT WHEN donation_cursor%NOTFOUND;

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

```

Results Explain Describe Saved SQL History

Member: Olivia Green - Total Donations: 120.75  
Member: Sophia Williams - Total Donations: 500  
Member: Emily Adams - Total Donations: 100  
Member: Daniel Brown - Total Donations: 75.25  
Member: James Carter - Total Donations: 250.5

Statement processed.

0.00 seconds



## 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
CURSOR event_cursor IS
SELECT Event_ID, Event_Name, Event_Date
FROM Event;

      TYPE Event_Record_Type IS RECORD (
Event_ID INT,
Event_Name VARCHAR2(255),
Event_Date DATE
);

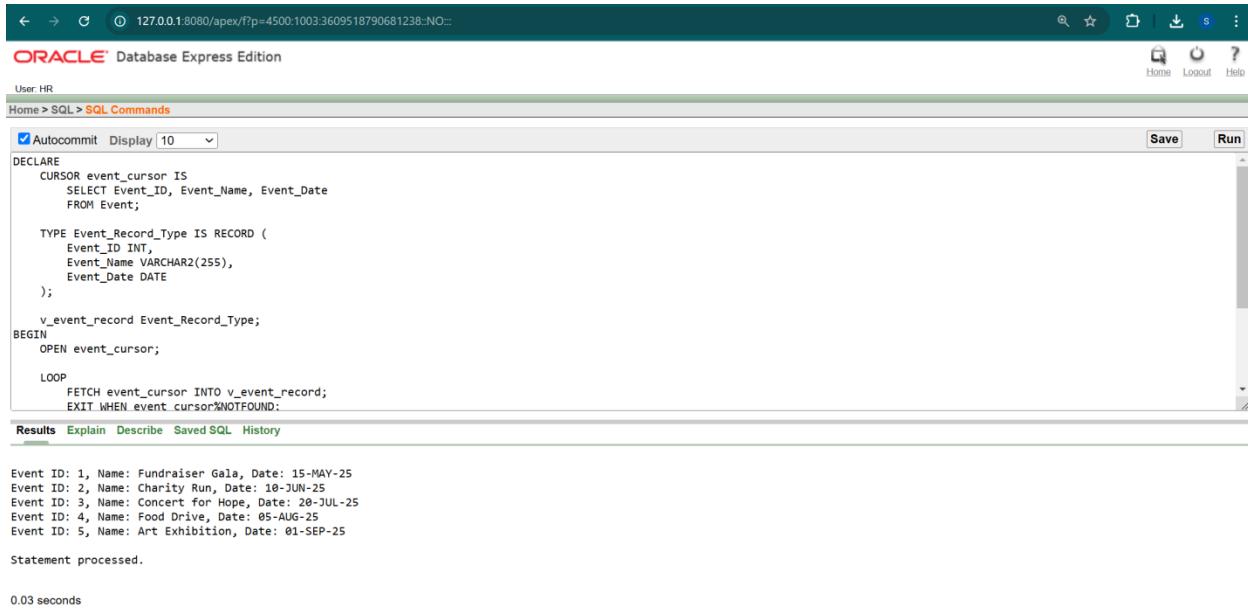
v_event_record Event_Record_Type;
BEGIN
OPEN event_cursor;

LOOP
FETCH event_cursor INTO v_event_record;
EXIT WHEN event_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;

CLOSE event_cursor;
END;

```



The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code is identical to the one above, but it includes a DBMS\_OUTPUT.PUT\_LINE statement to print the results directly from the procedure. The output window shows the following data:

```

Event ID: 1, Name: Fundraiser Gala, Date: 15-MAY-25
Event ID: 2, Name: Charity Run, Date: 10-JUN-25
Event ID: 3, Name: Concert for Hope, Date: 20-JUL-25
Event ID: 4, Name: Food Drive, Date: 05-AUG-25
Event ID: 5, Name: Art Exhibition, Date: 01-SEP-25

```

Below the output, a message states "Statement processed." and "0.03 seconds". At the bottom, the footer indicates "Language: en-us" and "Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved."



## 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;

127.0.0.1:8080/apex/f?p=4500:1003:3609518790681238::NO:::

ORACLE Database Express Edition

User HR
Home > SQL > SQL Commands

Save Run
Autocommit Display | 10 ▾
DECLARE
  CURSOR donation_cursor IS
    SELECT m.Name, SUM(d.Amount) AS Total_Donations
    FROM Member m
    JOIN Donation d ON m.Member_ID = d.Member_ID
    GROUP BY m.Name;

  TYPE Donation_Record_Type IS RECORD (
    Name VARCHAR2(255),
    Total_Donations DECIMAL(10,2)
  );

  v_donation_record Donation_Record_Type;
BEGIN
  OPEN donation_cursor;
  LOOP
    FETCH donation_cursor INTO v_donation_record;
    Member: Olivia Green - Total Donations: 120.75
    Member: Sophia Williams - Total Donations: 500
    Member: Emily Adams - Total Donations: 100
    Member: Daniel Brown - Total Donations: 75.25
    Member: James Carter - Total Donations: 250.5
  END LOOP;
  CLOSE donation_cursor;
  Statement processed.
  0.01 seconds

Language: en-us Application Express 2.1.0.00.39
Copyright © 1999, 2006, Oracle. All rights reserved.

```

## 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?

```

CREATE OR REPLACE TRIGGER track_donations_after_insert
AFTER INSERT ON Donation
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('Donation Added: ' || :NEW.Donor_Name || ' donated ' || 
:NEW.Amount);

```

```

END;

```

127.0.0.1:8080/apex/f?p=4500:1003:3609518790681238::NO::

**ORACLE® Database Express Edition**

User: HR

Home > SQL > SQL Commands

Autocommit Display: 10

CREATE OR REPLACE TRIGGER track\_donations\_after\_insert  
AFTER INSERT ON Donation  
FOR EACH ROW  
BEGIN  
DBMS\_OUTPUT.PUT\_LINE('Donation Added: ' || :NEW.Donor\_Name || ' donated ' || :NEW.Amount);  
END;

Save Run

Results Explain Describe Saved SQL History

Trigger created.

.07 seconds

Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.



## 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;

User: HR

Home > SQL > **SQL Commands**

Autocommit

```
CREATE OR REPLACE TRIGGER track_donations_after_update
AFTER UPDATE ON Donation
FOR EACH ROW
BEGIN
    -- Log the update
    DBMS_OUTPUT.PUT_LINE('Donation Updated: ' || :OLD.Donor_Name || ' donation amount changed from ' || :OLD.Amount || ' to ' || :NEW.Amount);
END;
```

**Results Explain Describe Saved SQL History**

Trigger created.

0.02 seconds

Application Express 2 1.0.0.30  
Copyright © 1999, 2006, Oracle. All rights reserved.



## 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;
```

```
END;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```
CREATE OR REPLACE TRIGGER log_activity
AFTER INSERT ON Donation
BEGIN
  DBMS_OUTPUT.PUT_LINE('Donation Inserted');
END;
```

The interface includes a toolbar with Home, Logout, and Help buttons, and a menu bar with Save and Run buttons. Below the code editor, there are tabs for Results, Explain, Describe, Saved SQL, and History.

Trigger created.

0.02 seconds

Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.



## 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;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The URL is 127.0.0.1:8080/apex/f?p=4500:1003:3609518790681238::NO:::. The page title is "ORACLE Database Express Edition". The user is "HR". The navigation bar includes "Home", "Logout", and "Help". The main area shows the following SQL code:

```
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;
```

Below the code, the results show:

```
Trigger created.
```

0.01 seconds

At the bottom right, it says "Application Express 2.1.0.00.39" and "Copyright © 1999, 2006, Oracle. All rights reserved."

## 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;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The URL is 127.0.0.1:8080/apex/f?p=4500:1003:360951879061238::NO:::. The page title is "ORACLE Database Express Edition". The navigation bar includes "Home", "Logout", and "Help". The main area shows the following SQL code:

```

CREATE OR REPLACE PACKAGE donation_pkg IS
  PROCEDURE add_donation(p_member_id INT, p_amount DECIMAL);
  FUNCTION get_total_donations(p_member_id INT) RETURN DECIMAL;
END donation_pkg;

```

Below the code, there are "Save" and "Run" buttons. The status bar at the bottom shows "Results Explain Describe Saved SQL History".

Package created.

0.02 seconds

Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.



## 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
WHERE Member_ID=p_member_id;

RETURN v_total;
END get_total_donations;
END donation_pkg;

```

The screenshot shows the Oracle Database Express Edition interface. The title bar reads "ORACLE Database Express Edition". The main window is titled "SQL Commands". The SQL code entered is:

```

CREATE OR REPLACE PACKAGE BODY donation_pkg IS
PROCEDURE add_donation(p_member_id INT, p_amount DECIMAL) IS
BEGIN
  INSERT INTO Donation (Member_ID, Amount, Donation_Date)
  VALUES (p_member_id, p_amount, SYSDATE);
END add_donation;

FUNCTION get_total_donations(p_member_id INT) RETURN DECIMAL IS
  v_total DECIMAL(10, 2);
BEGIN
  SELECT SUM(Amount) INTO v_total
  FROM Donation
  WHERE Member_ID = p_member_id;

  RETURN v_total;
END get_total_donations;
END donation_pkg;

```

Below the code, the results show "Package Body created." and "0.03 seconds".

Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.



## **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

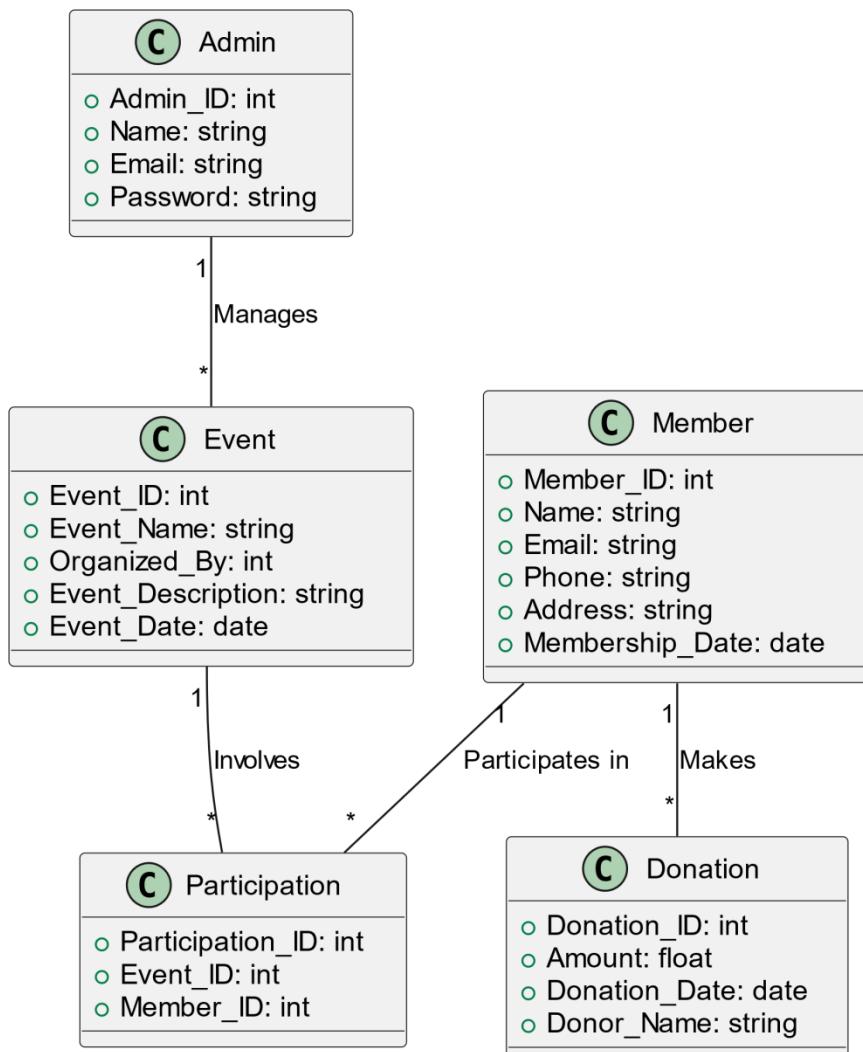
- **Admin:**
  - Create and manage events.
  - Manage member records.
  - Add new members.
  - Monitor financial transactions, including donations and expenses.
- **Member:**
  - View upcoming events.
  - Register and participate in events.
  - Make donations for events.

### **Technologies Used:**

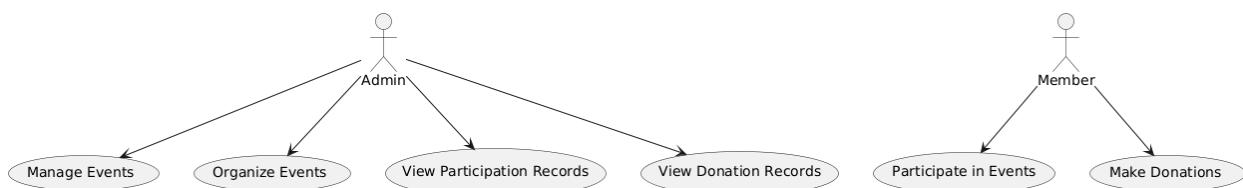
- **Frontend:** HTML, CSS
- **Backend:** Php
- **Database:** Oracle Database 10g

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.

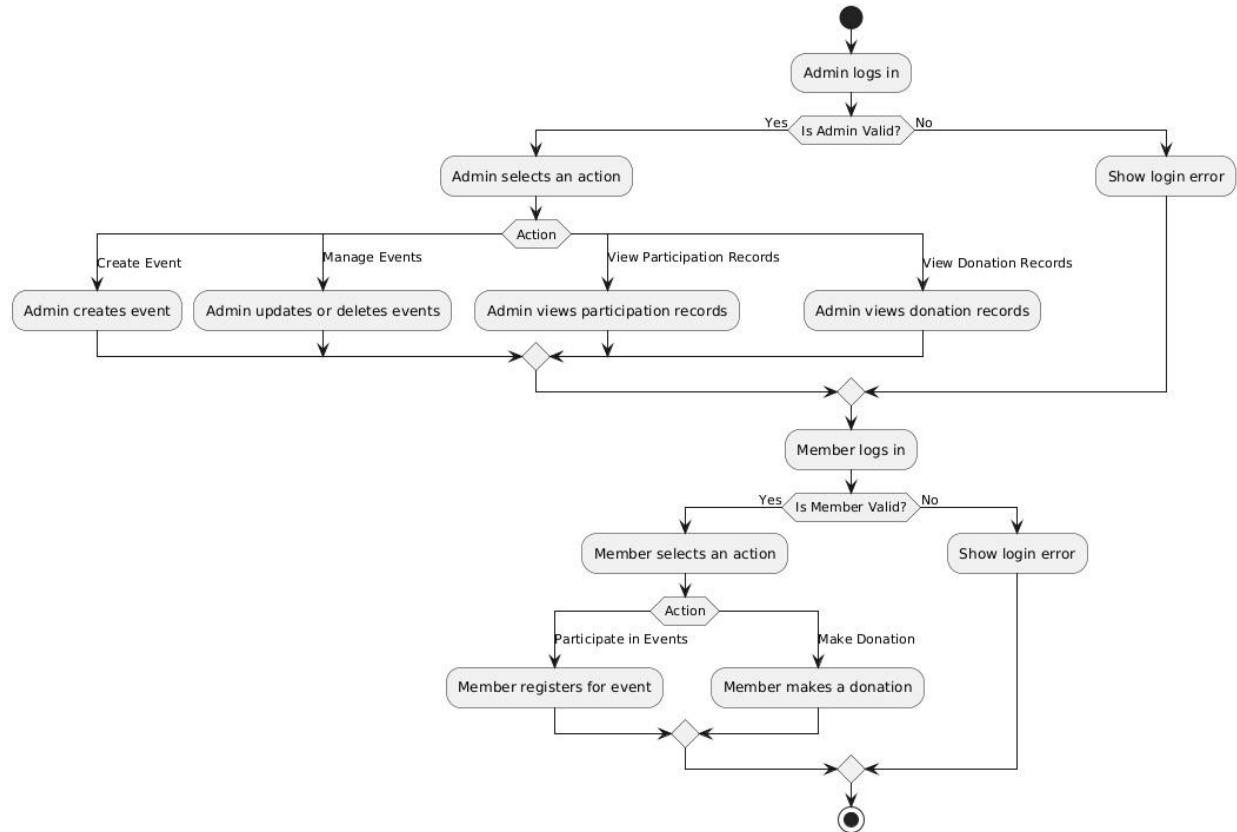
## Diagram



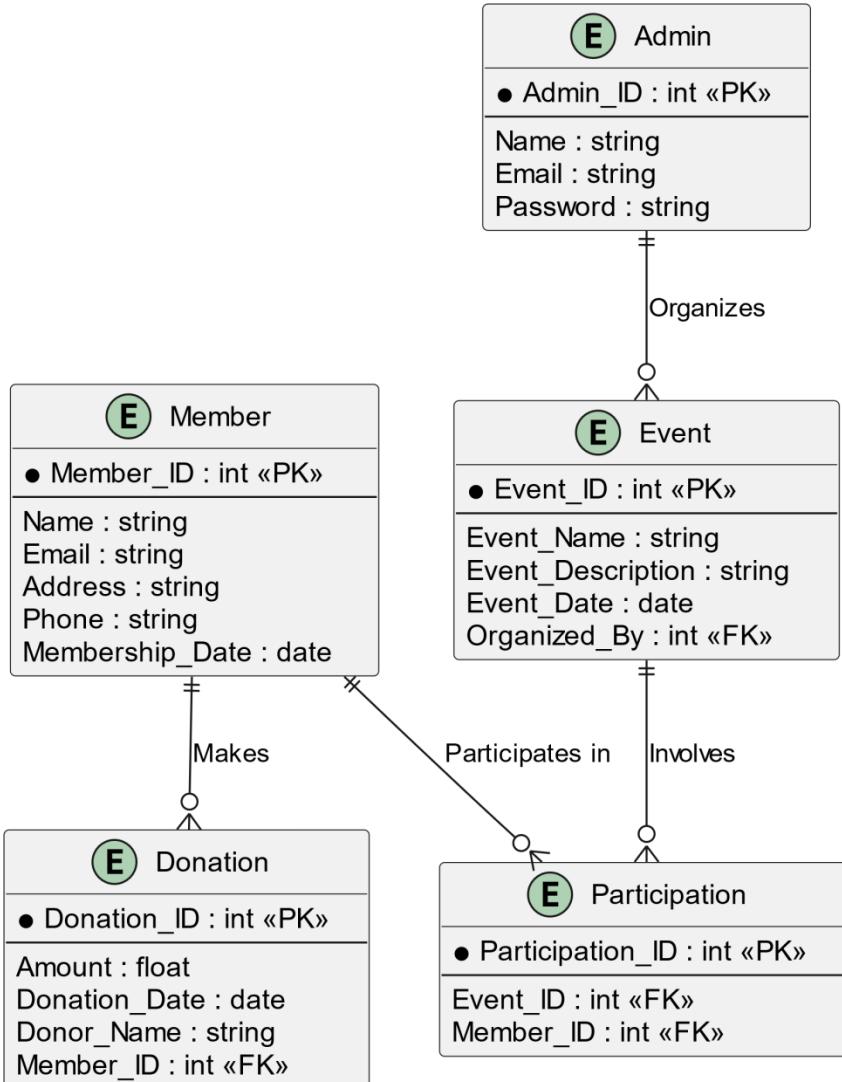
Class Diagram



Use Case Diagram



Activity Diagram



Schema Diagram

## User Interface

The screenshot shows a website layout for a club:

- Header:** Club Logo, Home, About Us, Join, Contact, Login/Signup.
- Content Area:**
  - About the Club:** A section with a "Learn More" button and a "Join Now" button.
  - Upcoming Events:** A section titled "Upcoming Events" showing "Event 1".

## 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

### Menu

- Home
- My Profile
- Events
- Members
- Finances
- Messages
- Reports
- Settings

## Dashboard

### Overview

Total Members  
120

Funds Raised  
\$15,000

Upcoming Events  
3

### Quick Actions

[Create Event](#)[Send Message](#)[Add Member](#)

## 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:**

**Date:**

**Time:**

# Membership Management

## Member Directory

Search:  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

# 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:

Type:



Category:

Description:

Add Transaction

## Budget Planning

Event or Cause:

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 Inread

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

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Club Landing Page</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>

    <header>
        <div class="logo">Club Logo</div>
        <nav>
            <ul>
                <li><a href="#">Home</a></li>
                <li><a href="#">About Us</a></li>
                <li><a href="#">Join</a></li>
                <li><a href="#">Contact</a></li>
            </ul>
        </nav>
        <button class="login-signup">Login/Signup</button>
    </header>

    <section class="main-banner">
        <h1>Join Us in Making a Difference!</h1>
        <button class="cta">Learn More</button>
        <button class="cta">Join Now</button>
    </section>

    <section id="about" class="about-club">
        <h2>About the Club</h2>
        <p>We are a group of passionate individuals dedicated to creating meaningful impacts in our community. Join us to make a difference together!</p>
        
    </section>

    <section id="events" class="upcoming-events">
        <h2>Upcoming Events</h2>
        <div class="event">
            <h3>Event 1</h3>
            <p>Date: January 15, 2024</p>
            <p>Description: A fun and engaging workshop on community development.</p>
        </div>
        <div class="event">
            <h3>Event 2</h3>
            <p>Date: February 10, 2024</p>
            <p>Description: Charity drive to support underprivileged families.</p>
        </div>
        <button class="view-all">View All Events</button>
    </section>
</body>
```

Code: Home page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dashboard</title>
    <link rel="stylesheet" href="styles2.css">
</head>
<body>

    <div class="dashboard-container">
        <aside class="sidebar">
            <h2>Menu</h2>
            <ul>
                <li><a href="#">Home</a></li>
                <li><a href="#">My Profile</a></li>
                <li><a href="#">Events</a></li>
                <li><a href="#">Members</a></li>
                <li><a href="#">Finances</a></li>
                <li><a href="#">Messages</a></li>
                <li><a href="#">Reports</a></li>
                <li><a href="#">Settings</a></li>
            </ul>
        </aside>
        <main class="main-panel">
            <header class="panel-header">
                <h1>Dashboard</h1>
            </header>

            <section class="overview">
                <h2>Overview</h2>
                <div class="stat">
                    <div class="stat">
                        <h3>Total Members</h3>
                        <p>120</p>
                    </div>
                    <div class="stat">
                        <h3>Funds Raised</h3>
                        <p>$15,000</p>
                    </div>
                    <div class="stat">
                        <h3>Upcoming Events</h3>
                        <p>3</p>
                    </div>
                </div>
            </section>

            <section class="quick-actions">
                <h2>Quick Actions</h2>
                <ul>
                    <li>...
```

Code: Admin Dashboard

The screenshot shows the Sublime Text interface with three tabs open: index3.html, index.html, and index2.html. The index3.html tab is active, displaying the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Profile</title>
    <link rel="stylesheet" href="styles3.css">
</head>
<body>
    <div class="profile-container">
        <section class="personal-info">
            <h1>My Profile</h1>
            <div class="info">
                
                <div>
                    <p><strong>Name:</strong> Mohammad rafiq</p>
                    <p><strong>Email:</strong> rafiq@example.com</p>
                    <p><strong>Phone:</strong> +880 123 456 7890</p>
                </div>
            </div>
        </section>
        <section class="membership-details">
            <h2>Membership Details:</h2>
            <p><strong>Role:</strong> Club Member</p>
            <p><strong>Join Date:</strong> December 05, 2023</p>
            <p><strong>Due Date:</strong> March 5, 2023</p>
            <p><strong>Dues:</strong> Paid</p>
            <button class="edit-profile">Edit Profile</button>
        </section>
        <section class="activity-log">
            <h2>Activity Log</h2>
            <ul>
                <li>Joined the "Tree Plantation Drive" on Dec 1, 2023.</li>
                <li>Attended "Leadership Training Workshop".</li>
                <li>Messaged about member registration updates.</li>
                <li>Contributed to "Community Clean-Up Project".</li>
            </ul>
        </section>
    </div>
</body>
</html>
```

## Code : Member profile

The screenshot shows the Sublime Text interface with three tabs open: index3.html, index4.html, and index.html. The index4.html tab is active, displaying the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Event Management</title>
    <link rel="stylesheet" href="styles4.css">
</head>
<body>
    <header class="header">
        <div class="container">
            <h1>Event Management</h1>
            <p>Plan, manage, and track your events with ease.</p>
        </div>
    </header>
    <section class="event-list">
        <div class="container">
            <h2>Upcoming Events:</h2>
            <div class="event-card">
                <div class="event-card">
                    <h3>Fundraising Workshop</h3>
                    <p><strong>Date:</strong> December 20, 2023</p>
                    <p><strong>Time:</strong> 10:00 AM - 2:00 PM</p>
                    <button class="view-details">View Details</button>
                </div>
                <div class="event-card">
                    <h3>Annual Club Meeting</h3>
                    <p><strong>Date:</strong> January 15, 2024</p>
                    <p><strong>Time:</strong> 5:00 PM - 8:00 PM</p>
                    <button class="view-details">View Details</button>
                </div>
                <div class="event-card">
                    <h3>Tree Plantation Drive</h3>
                    <p><strong>Date:</strong> December 25, 2023</p>
                    <p><strong>Time:</strong> 9:00 AM - 12:00 PM</p>
                    <button class="view-details">View Details</button>
                </div>
            </div>
        </div>
    </section>
    <section class="create-event">
        <div class="container">
            <h2>Create New Event:</h2>
            <form>
                <div class="form-group">
                    <label for="title">Event Title:</label>
                    <input type="text" id="title" placeholder="Enter event title" required>
                </div>
            </div>
        </section>
    </body>
</html>
```

## 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:

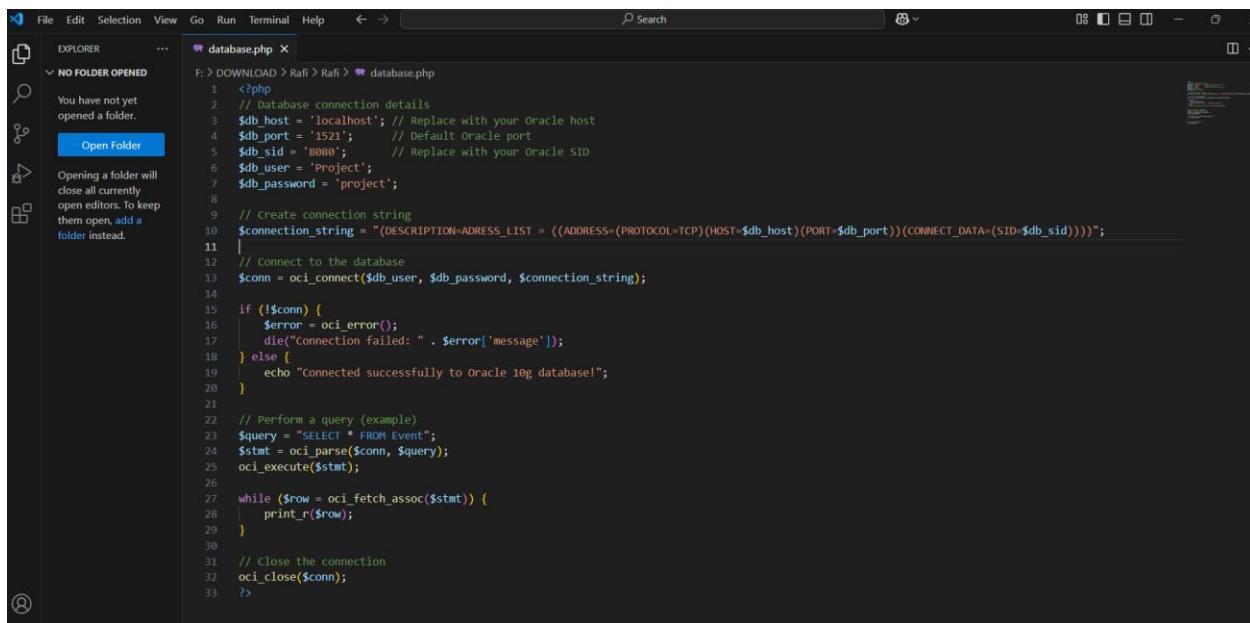
1. Installed Oracle 10g and configured the database.
2. Used php to establish a connection.
3. Attempted to configure connection settings using Oracle Client, SQL\*Plus, or ODBC/JDBC.
4. Maybe We have problem with the credentials, hostname, port, and service name in the connection string.
5. For Debugging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

### **3. Challenges Faced in Database Connection**

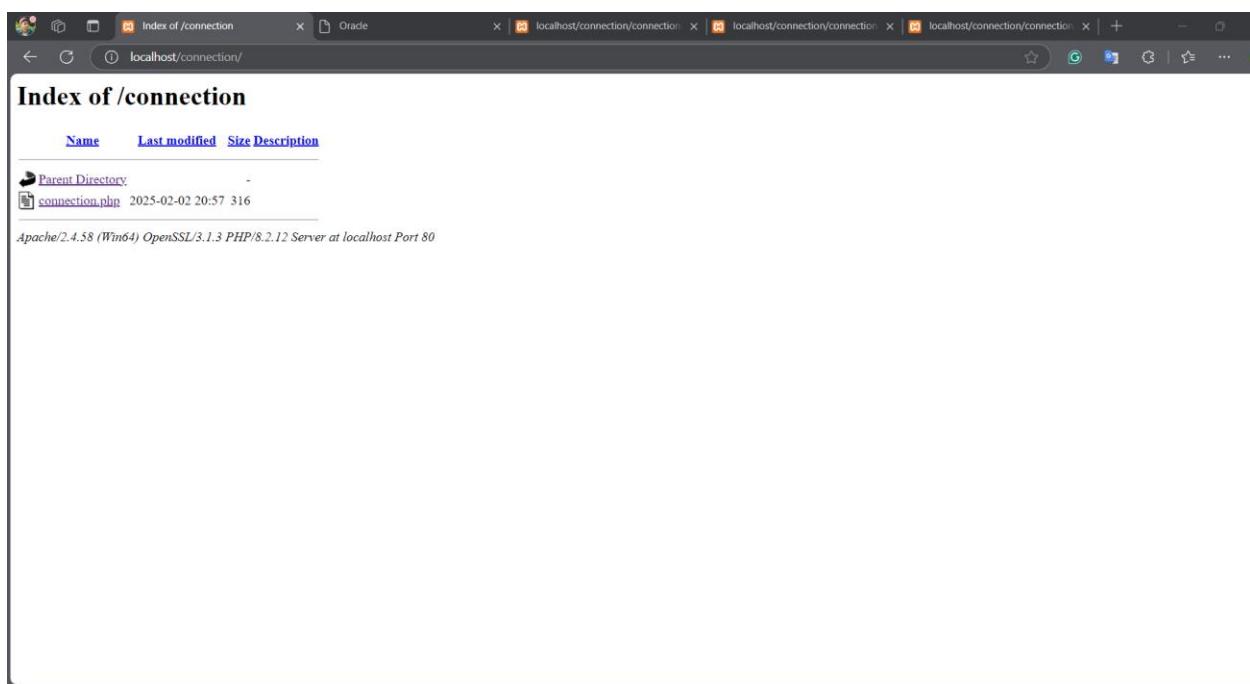
#### **Challenges Faced**

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.



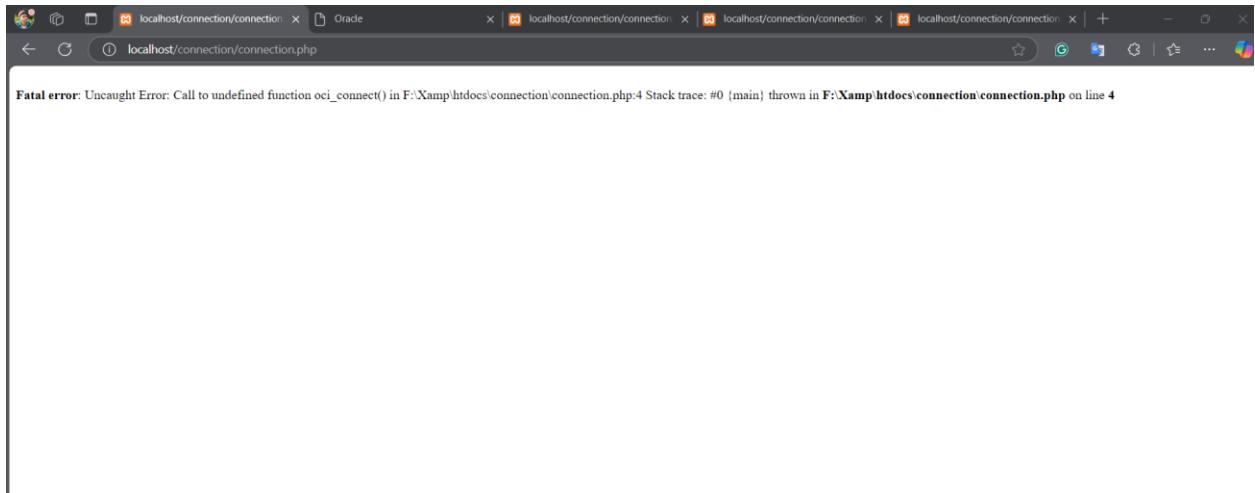
```
F:\> DOWNLOAD > Rafi > Rafi > database.php
1  <?php
2  // Database connection details
3  $db_host = 'localhost'; // Replace with your Oracle host
4  $db_port = '1521'; // Default Oracle port
5  $db_sid = '8888'; // Replace with your Oracle SID
6  $db_user = 'Project';
7  $db_password = 'project';
8
9  // Create connection string
10 $connection_string = "((DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=$db_host)(PORT=$db_port))(CONNECT_DATA=(SID=$db_sid)))";
11 |
12 // Connect to the database
13 $conn = oci_connect($db_user, $db_password, $connection_string);
14
15 if (!$conn) {
16     $error = oci_error();
17     die("Connection failed: " . $error['message']);
18 } else {
19     echo "Connected successfully to Oracle 10g database!";
20 }
21
22 // Perform a query (example)
23 $query = "SELECT * FROM Event";
24 $stmt = oci_parse($conn, $query);
25 oci_execute($stmt);
26
27 while ($row = oci_fetch_assoc($stmt)) {
28     print_r($row);
29 }
30
31 // Close the connection
32 oci_close($conn);
33 ?>
```



## Index of /connection

Name	Last modified	Size	Description
Parent Directory			
<a href="#">connection.php</a>	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;

```

ORACLE Database Express Edition

User HR

Home > SQL > SQL Commands

Autocommit

```

DECLARE
    v_member_id INT := 1; -- Example member ID
    v_amount DECIMAL(10, 2) := 100.00; -- Donation amount
    v_donation_date DATE := SYSDATE; -- Current date
BEGIN
    BEGIN
        -- Insert a donation record using a sequence for Donation_ID
        INSERT INTO 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; -- Commit if the insert is successful
    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;

```

Results Explain Describe Saved SQL History

An unexpected error occurred: ORA-04098: trigger 'HR.UPDATE\_MEMBER\_STATUS' is invalid and failed re-validation

1 row(s) inserted.

0.03 seconds

Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.



## 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_id INT := 1; -- Example member ID
v_name VARCHAR2(255) := 'John Smith'; -- New name
v_email VARCHAR2(255) := 'john.smith@example.com'; -- New email
BEGIN
BEGIN
UPDATERMember
SET Name =v_name, Email =v_email
WHERE Member_ID=v_member_id;

IF SQL%ROWCOUNT =0THEN

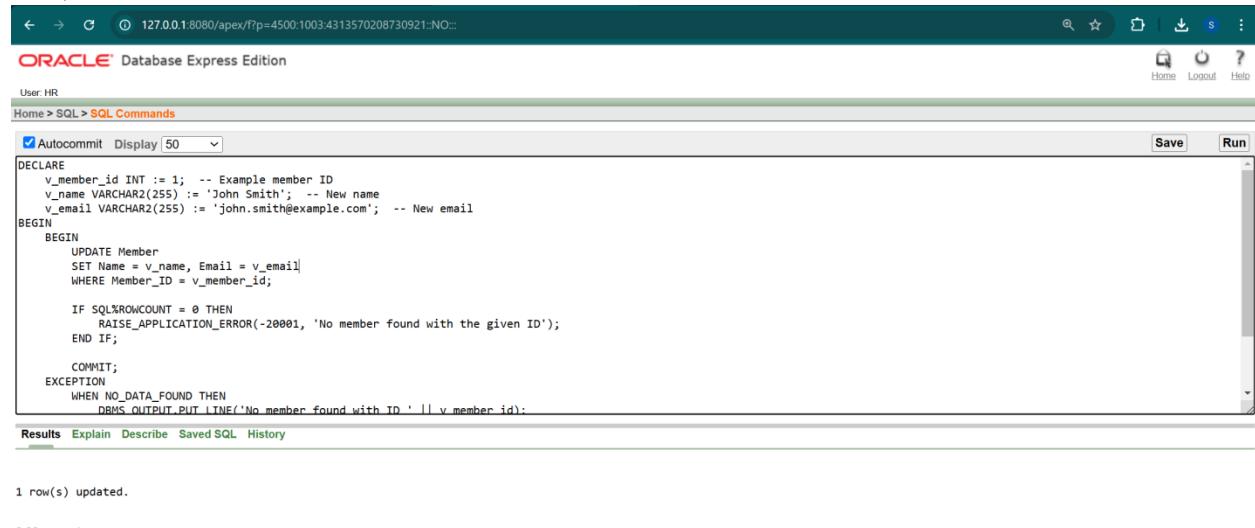
```

```

        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;

```



The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands tab, a PL/SQL block is run. The code declares variables, updates a member record, and handles errors using RAISE\_APPLICATION\_ERROR and DBMS\_OUTPUT.PUT\_LINE. The results pane shows "1 row(s) updated." and "0.00 seconds". The status bar at the bottom right indicates "Application Express 2.1.0.00.39" and "Copyright © 1999, 2006, Oracle. All rights reserved."

## 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;

```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a PL/SQL block is run. The output shows the message 'Row locked for update' and '1 row(s) updated.' indicating that the row was successfully locked.

```

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;
  END IF;
END;

```

Results Explain Describe Saved SQL History

Row locked for update  
1 row(s) updated.  
0.01 seconds

Language: en-US Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.



## 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
INTO v_donation_amount
FROM Donation
WHERE Member_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
WHERE Member_ID=v_member_id;

COMMIT;
END;

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code entered is a PL/SQL block that declares variables, begins a transaction, locks a row, and updates the donation amount. The results show the update was successful.

```

User HR
Home > SQL > SQL Commands
Autocommit Display 50 Save Run
DECLARE
v_member_id INT :=1;
v_donation_amount DECIMAL(10, 2) := 200.00;
BEGIN
-- Lock member's record implicitly for update
SELECT Amount
INTO v_donation_amount
FROM Donation
WHERE Member_ID = v_member_id
FOR UPDATE; -- Lock the member's donation row implicitly

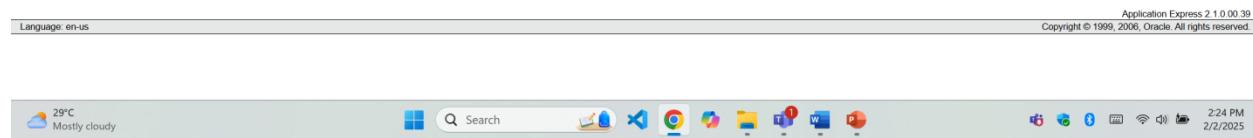
-- Process donation or update donation history
UPDATE Donation
SET Amount = v_donation_amount + 100 -- Example donation update
WHERE Member_ID = v_member_id;

COMMIT;
END;

```

**Results**

Donation Updated: John Doe donation amount changed from 100 to 200  
1 row(s) updated.  
0.02 seconds



### 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
INSERT INTO Donation (Donation_ID, Amount, Donation_Date, Donor_Name,
Member_ID)
VALUES (Donation_SEQ.NEXTVAL, 500.00, SYSDATE, 'Jane Doe', 2);

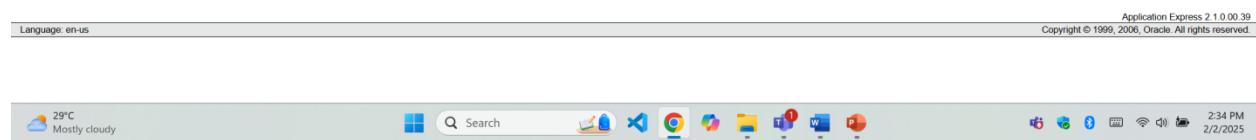
COMMIT;
END;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code in the editor is identical to the one above. The results section shows the output of the executed query:

```
Donation Added: Jane Doe donated 500
Total amount donated: 1646.5
Donation Inserted

1 row(s) inserted.

0.01 seconds
```



## 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
SET Event_Name='Updated Charity Run'
WHERE Event_ID=2;

COMMIT;
END;
```

The screenshot shows the Oracle Database Express Edition interface. The URL is 127.0.0.1:8080/apex/f?p=4500:1003:4313570208730921::NO:::. The page title is "ORACLE Database Express Edition". The user is "User HR". The navigation bar includes "Home", "Logout", and "Help". The main area shows a PL/SQL block:

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

  -- Proceed with updating the event
  UPDATE Event
  SET Event_Name = 'Updated Charity Run'
  WHERE Event_ID = 2;

  COMMIT;
END;
```

Buttons "Save" and "Run" are visible. Below the code, the results are shown:

1 row(s) updated.

0.00 seconds

Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.



## Relational Algebra:

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

Ans:  $\sigma \text{donation\_amount} > 500$  (Donations)

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

Ans:  $\Pi(\text{event\_name}, \text{event\_date})$  (Events)

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

Ans:  $\gamma \text{member\_id}, \text{SUM}(\text{donation\_amount})$  (Donations)

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

Ans:  $\Pi(\text{event\_id}) \text{ (Events)} \cap \Pi(\text{event\_id}) \text{ (Donations)}$

5. Rename the column "event\_date" in the Events table to "date\_of\_event". (Rename)

Ans:  $\rho\{\text{Events}(\text{event\_id}, \text{event\_name}, \text{date\_of\_event}, \text{location})\} \text{ (Events)}$

**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.