PROJECT RELATIONAL SCHEMA
G5_T12

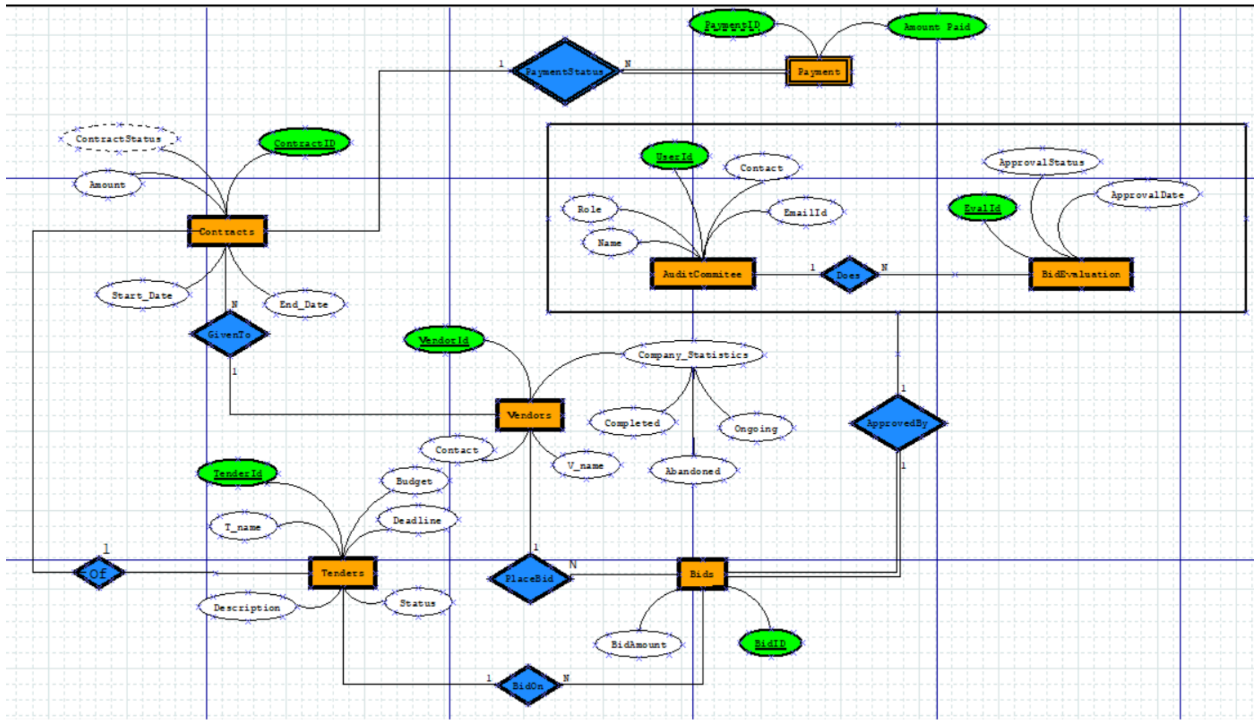
Members:

Tirth K Gandhi(202301413)

Sheel Shah(202301420)

Manit Shah(202301425)

UPDATED ER DIAGRAM:



Relational Schema:

1. Tender(TenderID,T_Name,Description,Status,Budget,Deadline)

//PK: TendorID

2. Vendor(VendorID,V_Name,Contact,CompletedContracts,Ongoin-gContracts,AbandonedContracts)
   //PK: VendorID
3. Bids(BidID,BidAmount,TenderId,VendorId,EvaluatedBy,EvalID)
   //PK: BidID
   //FK: {EvaluatedBy,EvalID} References to AuditCommittee and BidEval Relationship
   //FK: TenderId References to Tender
   //FK: VendorId References to Vendor
4. AuditCommittee(UserID,Contact,EmailID,Role,Name)
   //PK: UserID
5. BidEval(EvalID,ApprovalDate,ApprovalStatus,UserID)
   //PK: EvalID
   //FK: UserID References to AuditCommittee
6. Contract(ContractID,TendorId,VendorId,ContractStatus,Start_Dat--e,End_Date,Amount)
   //PK: ContractID
   //FK: TenderId References to Tender
   //FK: VendorId References to Vendor
7. Payments(ContractID,PaymentID,AmountPaid,PaymentDate)
   //PK: {ContractID,PaymentID}
   //FK: ContractID References to Contract

<u>**Assumptions Taken:**</u>

❖ VendorID is Registration Number
❖ Contact and EmailID are always unique and single no person can get a duplicate email or contact.
❖ TenderName and VendorName need not be unique.
❖ Different Vendors Bid different amounts. For Less complication we have assumed this since if the lowest bid is by more than one vendor it would create conflicts. This is our Future improvements we would keep in mind.

FD Set – Entity-Wise

Entity: Tender

TenderId —>{ T_name,Description,Budget,Deadline,Status}

Min Fd set : TenderId —>{ T_name,Description,Budget,Deadline,Status}

Key: TenderId

The relations are in BCNF as all the elements of the relation can be determined with the help of the key.

Entity: Vendor

VendorId —> {V-Name, Contact, Completed_Contracts, Ongoing_Contracts, Abandoned_Contracts}

Min Fd set :

VendorId —> {V-Name, Contact, Completed_Contracts, Ongoing_Contracts, Abandoned_Contracts}

Key: VendorId

The relations are in BCNF as all the elements of the relation can be determined with the help of the key.

Entity: Bids

BidId —> {BidAmount,EvaluatedBy, EvalId,TenderID,VendorID}

EvalId ----->{BidAmount,EvaluatedBy, EvalId,TenderID,VendorID}

Min Fd set :

EvalId —> BidId

BidId —> {BidAmount,EvaluatedBy,TenderID,VendorID,EvalId}

Key: {BidId},{EvalId}

The relations are in BCNF as all the elements of the relation can be determined with the help of either of the keys.

Entity: AuditCommittee

UserID —> {Contact, EmailID, Role, Name}

EmailID —> {Contact, UserID, Role, Name}

Contact —> {UserId, EmailID, Role, Name}

Min Fd set :

Contact —> EmailID

EmailID —> UserID

UserID —> {Role,Name,EmailID,Contact}

Key: {UserID},{EmailID},{ContactID}

The relations are in BCNF as all the elements of the relation can be determined with the help of all the keys of the relation.

Entity: BidEval

EvalID —> {ApprovalDate,ApprovalStatus,UserID}

Min Fd set :

EvalID —> {ApprovalDate,ApprovalStatus,UserID}

Key: EvalID

The relations are in BCNF as all the elements of the relation can be determined with the help of the key of the relation.

Entity: Contract

ContractID —> {ContractStatus,Start_Date,End_Date,Amount,TenderID,VendorID}

TenderID —> ContractID

Min Fd set :

ContractID —> {ContractStatus,StartDate,EndDate,Amount,TenderID,VendorID}

TenderID —> ContractID

Key: {ContractID}

BCNF Prove: The relation for TenderID to ContractID is **not in BCNF** as TenderID is not the key for this entity.

We could have converted this to BCNF by using the **BCNF Decomposition algorithm** by adding a new entity TendorContract having key {TendorID,VendorID} but we chose not as when we tried to access data like tenders for a particular vendor, vendor for a particular tender and so on we had to perform multiple joins which could have jumbled the data at some point and took a lot time to execute the query. Hence it did not seem feasible to decompose into BCNF.

However, the relations of the FD set are in **3NF** as in the relation TenderID -> ContractID, ContractID is a prime attribute.

Entity: Payments

{ContractID,PaymentID} —> AmountPaid,PaymentDate

Min Fd set :

{ContractID,PaymentID} —> AmountPaid,PaymentDate

Key: {ContractID,PaymentID}

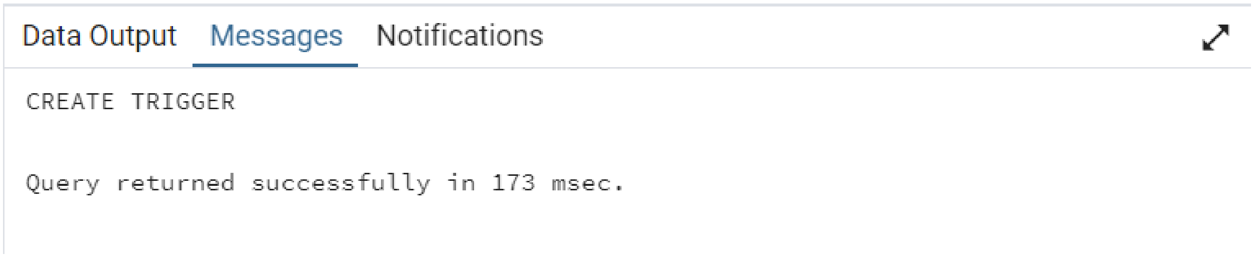
The relations are in BCNF as all the elements of the relation can be determined with the help of the key of the relation.

DDL Scripts+Triggers

```
CREATE SCHEMA tender;
SET search_path TO tender;

-- 1. Tender Table
CREATE TABLE Tender (
    TenderID SERIAL PRIMARY KEY,
    T_Name VARCHAR(100),
    Description TEXT,
    Status VARCHAR(20),
    Budget NUMERIC(12, 2),
    Deadline DATE
);

-- 2. Vendor Table
CREATE TABLE Vendor (
    VendorID SERIAL PRIMARY KEY,
    V_Name VARCHAR(100),
    Contact VARCHAR(15),
    CompletedContracts INT DEFAULT 0,
    OngoingContracts INT DEFAULT 0,
    AbandonedContracts INT DEFAULT 0,
    IsRedFlagged BOOLEAN DEFAULT FALSE,
    IsBlocked BOOLEAN DEFAULT FALSE
);

-- 3. AuditCommittee Table
CREATE TABLE AuditCommittee (
    UserID SERIAL PRIMARY KEY,
    Contact VARCHAR(15),
    EmailID VARCHAR(100),
    Role VARCHAR(50),
    Name VARCHAR(100)
);
```

```sql
-- 4. BidEval Table (no manual insert - trigger controlled)
CREATE TABLE BidEval (
    EvalID SERIAL PRIMARY KEY,
    ApprovalDate DATE DEFAULT CURRENT_DATE,
    ApprovalStatus VARCHAR(20),
    FOREIGN KEY (UserID) INT REFERENCES AuditCommittee(UserID)
);

-- 5. Bids Table
CREATE TABLE Bids (
    BidID SERIAL PRIMARY KEY,
    BidAmount NUMERIC(12, 2),
    FOREIGN KEY (TenderID) INT REFERENCES Tender(TenderID),
    FOREIGN KEY (VendorID) INT REFERENCES Vendor(VendorID),
    FOREIGN KEY (EvalID) INT REFERENCES BidEval(EvalID),
    FOREIGN KEY (EvaluatedBy) INT REFERENCES AuditCommittee(UserID)
);

-- 6. Contract Table
CREATE TABLE Contract (
    ContractID SERIAL PRIMARY KEY,
    FOREIGN KEY (TenderID) INT REFERENCES Tender(TenderID),
    FOREIGN KEY (VendorID) INT REFERENCES Vendor(VendorID),
    ContractStatus VARCHAR(30),
    Start_Date DATE,
    End_Date DATE,
    Amount NUMERIC(12, 2)
);

--7. Payments Table (Weak Entity)
CREATE TABLE Payments (
    PaymentID SERIAL,
    ContractID INT,
    AmountPaid NUMERIC(12, 2),
    PaymentDate DATE,
    PRIMARY KEY (PaymentID, ContractID),
    FOREIGN KEY (ContractID) REFERENCES Contract(ContractID)
);
```

```sql
CREATE OR REPLACE FUNCTION handle_redflag_vendor()
RETURNS TRIGGER AS $$
BEGIN
    -- Check if vendor's abandoned contracts have reached or exceeded 2
    IF NEW.AbandonedContracts >= 2 THEN
        -- Only update if the vendor is not already red-flagged or blocked
        IF NOT EXISTS (SELECT 1 FROM Vendor WHERE VendorID = NEW.VendorID
AND IsRedFlagged = TRUE AND IsBlocked = TRUE) THEN
            UPDATE Vendor
            SET IsRedFlagged = TRUE, IsBlocked = TRUE
            WHERE VendorID = NEW.VendorID;

            -- Delete all ongoing bids of the vendor
            DELETE FROM Bids
            WHERE VendorID = NEW.VendorID;
        END IF;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trg_redflag_check
AFTER INSERT OR UPDATE ON Vendor
FOR EACH ROW
EXECUTE PROCEDURE handle_redflag_vendor();


CREATE TRIGGER trg_redflag_check
AFTER INSERT OR UPDATE ON Vendor
FOR EACH ROW
EXECUTE PROCEDURE handle_redflag_vendor();

-- Trigger 2: Automatically create a BidEval entry based on vendor status
CREATE OR REPLACE FUNCTION auto_evaluate_bid()
RETURNS TRIGGER AS $$
DECLARE
    is_blocked BOOLEAN;
    audit_user INT;
    new_eval_id INT;
    approval TEXT;
BEGIN
    SELECT IsBlocked INTO is_blocked
    FROM Vendor
    WHERE VendorID = NEW.VendorID;

    SELECT UserID INTO audit_user
    FROM AuditCommittee
    ORDER BY RANDOM()
    LIMIT 1;

    IF is_blocked THEN
        approval := 'Rejected';
    ELSE
        approval := 'Approved';
    END IF;

    INSERT INTO BidEval (ApprovalStatus, UserID)
    VALUES (approval, audit_user)
    RETURNING EvalID INTO new_eval_id;

    NEW.EvalID := new_eval_id;
    NEW.EvaluatedBy := audit_user;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_auto_eval_bid
BEFORE INSERT ON Bids
FOR EACH ROW
EXECUTE PROCEDURE auto_evaluate_bid();
```

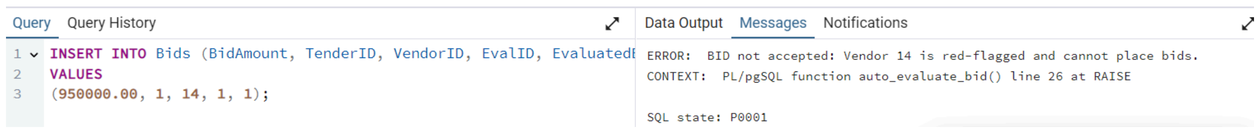

Data Output Messages Notifications

```
CREATE TRIGGER

Query returned successfully in 173 msec.
```

INSERT STATEMENTS:

```
-- Inserting into Tender Table
INSERT INTO Tender (T_Name, Description, Status, Budget, Deadline)
VALUES
('Tender 1', 'Tender for construction of office building', 'Open',
1000000.00, '2025-12-31'),
('Tender 2', 'Tender for renovation of office', 'Closed', 500000.00,
'2025-06-30'),
('Tender 3', 'Tender for road maintenance', 'Open', 300000.00,
'2025-08-15'),
('Tender 4', 'Tender for electrical wiring installation', 'Open',
200000.00, '2025-07-31'),
('Tender 5', 'Tender for IT infrastructure setup', 'Closed', 800000.00,
'2025-05-01');

-- Inserting into Vendor Table
INSERT INTO Vendor (V_Name, Contact, CompletedContracts, OngoingContracts,
AbandonedContracts)
VALUES
('Vendor A', '1234567890', 5, 2, 0),
('Vendor B', '2345678901', 3, 1, 1),
('Vendor C', '3456789012', 10, 0, 0),
('Vendor D', '4567890123', 7, 3, 2),
('Vendor E', '5678901234', 2, 1, 0);

-- Inserting into AuditCommittee Table
INSERT INTO AuditCommittee (Contact, EmailID, Role, Name)
VALUES
('9876543210', 'john.doe@example.com', 'Chairperson', 'John Doe'),
('8765432109', 'jane.smith@example.com', 'Member', 'Jane Smith'),
('7654321098', 'mark.jones@example.com', 'Member', 'Mark Jones'),
('6543210987', 'lucy.brown@example.com', 'Member', 'Lucy Brown'),
('5432109876', 'alice.white@example.com', 'Member', 'Alice White');

-- Inserting into Bids Table
INSERT INTO Bids (BidAmount, TenderID, VendorID, EvalID, EvaluatedBy)
VALUES
(950000.00, 1, 1, 1, 1),
(400000.00, 2, 2, 2, 2),
(280000.00, 3, 3, 3, 3),
(190000.00, 4, 4, 4, 4),
(750000.00, 5, 5, 5, 5),
(600000.00, 1, 3, 3, 1),
(500000.00, 2, 4, 4, 2),
(350000.00, 3, 2, 2, 3),
(220000.00, 4, 5, 5, 4),
(800000.00, 5, 1, 1, 5);

-- Inserting into Contract Table
INSERT INTO Contract (TenderID, VendorID, ContractStatus, Start_Date,
End_Date, Amount)
VALUES
(2, 12, 'Completed', '2025-03-01', '2025-06-01', 400000.00),
(5, 15, 'Completed', '2025-02-01', '2025-05-01', 750000.00),
(2, 14, 'Completed', '2025-01-01', '2025-04-01', 500000.00),
(4, 15, 'Completed', '2025-03-15', '2025-06-15', 220000.00);

-- Inserting into Payments Table
INSERT INTO Payments (ContractID, AmountPaid, PaymentDate)
VALUES
(21, 500000.00, '2025-05-15');
```

Trigger Function Output:(Used 2 Triggers)



Query Query History

```
INSERT INTO Bids (BidAmount, TenderID, VendorID, EvalID, Evaluated
VALUES
(950000.00, 1, 14, 1, 1);
```

Data Output Messages Notifications

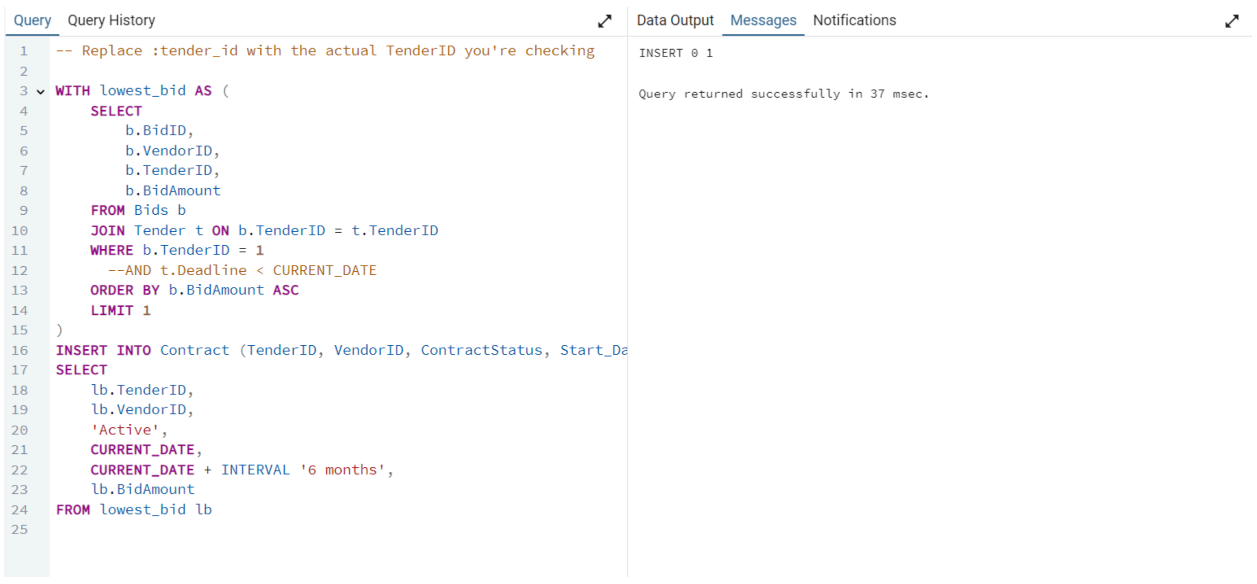
```
ERROR:  BID not accepted: Vendor 14 is red-flagged and cannot place bids.
CONTEXT:  PL/pgSQL function auto_evaluate_bid() line 26 at RAISE

SQL state: P0001
```

SOME QUERIES USEFUL FOR FUNCTIONING OF OUR MODEL

TOP Queries Extremely Useful:

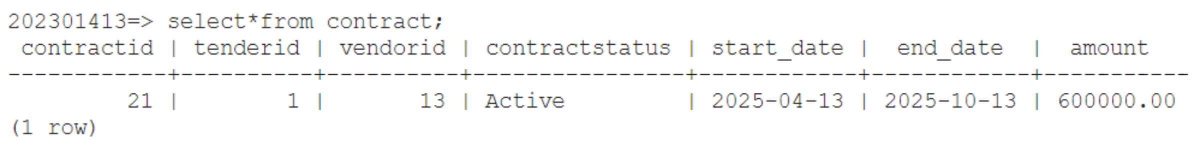
1. Select Lowest Bidder for a tender and give it the contract if deadline is over:

```
-- Replace :tender_id with the actual TenderID you're checking

WITH lowest_bid AS (
    SELECT
        b.BidID,
        b.VendorID,
        b.TenderID,
        b.BidAmount
    FROM Bids b
    JOIN Tender t ON b.TenderID = t.TenderID
    WHERE b.TenderID = :tender_id --Replace tendered with whatever you are
c                                  --checking
      --AND t.Deadline < CURRENT_DATE use only if doing after real
--deadline
    ORDER BY b.BidAmount ASC
    LIMIT 1
)
INSERT INTO Contract (TenderID, VendorID, ContractStatus, Start_Date,
End_Date, Amount)
SELECT
    lb.TenderID,
    lb.VendorID,
    'Active',
    CURRENT_DATE,
    CURRENT_DATE + INTERVAL '6 months',
    lb.BidAmount
FROM lowest_bid lb
```



```
-- Replace :tender_id with the actual TenderID you're checking

WITH lowest_bid AS (
    SELECT
        b.BidID,
        b.VendorID,
        b.TenderID,
        b.BidAmount
    FROM Bids b
    JOIN Tender t ON b.TenderID = t.TenderID
    WHERE b.TenderID = 1
      --AND t.Deadline < CURRENT_DATE
    ORDER BY b.BidAmount ASC
    LIMIT 1
)
INSERT INTO Contract (TenderID, VendorID, ContractStatus, Start_Da
SELECT
    lb.TenderID,
    lb.VendorID,
    'Active',
    CURRENT_DATE,
    CURRENT_DATE + INTERVAL '6 months',
    lb.BidAmount
FROM lowest_bid lb
```

```
INSERT 0 1

Query returned successfully in 37 msec.
```

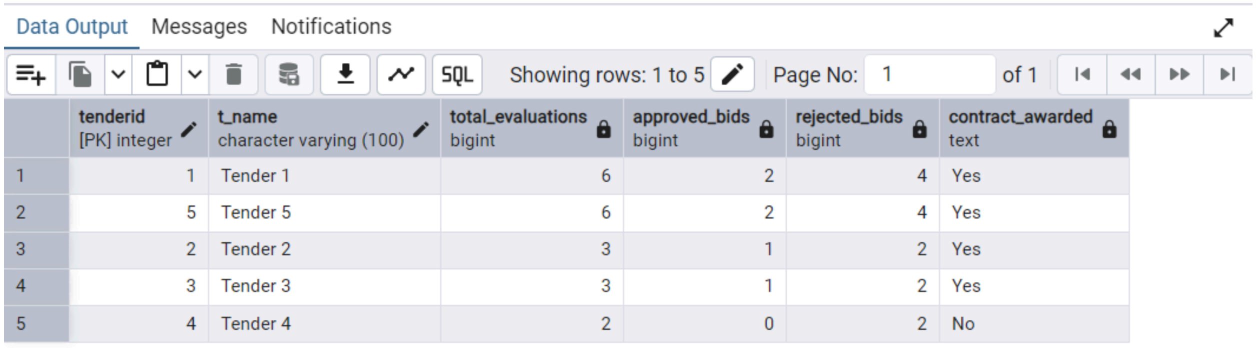


```
202301413=> select*from contract;
 contractid | tenderid | vendorid | contractstatus | start_date |  end_date  |  amount
------------+----------+----------+----------------+------------+------------+-----------
         21 |        1 |       13 | Active         | 2025-04-13 | 2025-10-13 | 600000.00
(1 row)
```

2. Tender Summary And Report

```
SELECT
    t.TenderID,
    t.T_Name,
    COUNT(be.EvalID) AS Total_Evaluations,
    SUM(CASE WHEN be.ApprovalStatus = 'Approved' THEN 1 ELSE 0 END) AS
Approved_Bids,
    SUM(CASE WHEN be.ApprovalStatus = 'Rejected' THEN 1 ELSE 0 END) AS
Rejected_Bids,
    CASE
        WHEN EXISTS (
            SELECT 1 FROM Contract c WHERE c.TenderID = t.TenderID
        ) THEN 'Yes' ELSE 'No'
    END AS Contract_Awarded
FROM Tender t
LEFT JOIN Bids b ON t.TenderID = b.TenderID
LEFT JOIN BidEval be ON b.EvalID = be.EvalID OR be.EvalID IN (
    SELECT EvalID FROM BidEval WHERE EvalID NOT IN (SELECT EvalID FROM
Bids)
)
```

```
GROUP BY t.TenderID, t.T_Name
ORDER BY Total_Evaluations DESC;
```



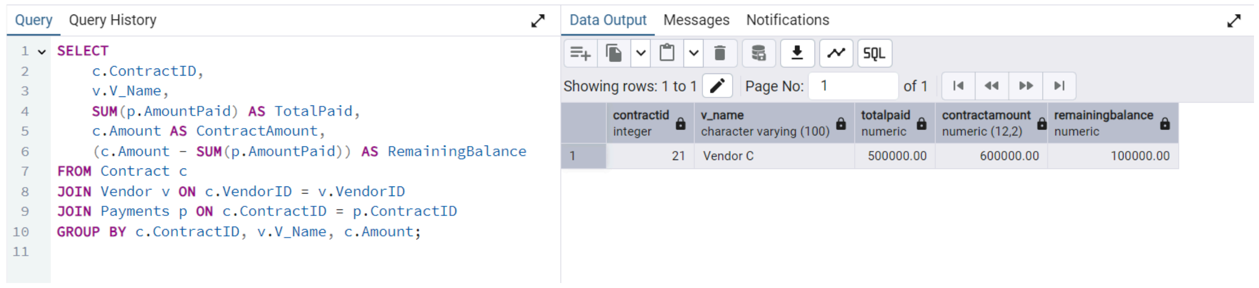
Data Output Messages Notifications

Showing rows: 1 to 5 Page No: 1 of 1

	tenderid [PK] integer	t_name character varying (100)	total_evaluations bigint	approved_bids bigint	rejected_bids bigint	contract_awarded text
1	1	Tender 1	6	2	4	Yes
2	5	Tender 5	6	2	4	Yes
3	2	Tender 2	3	1	2	Yes
4	3	Tender 3	3	1	2	Yes
5	4	Tender 4	2	0	2	No

3. Payment Status and Remaining Payment Details

```
SELECT
    c.ContractID,
    v.V_Name,
    SUM(p.AmountPaid) AS TotalPaid,
    c.Amount AS ContractAmount,
    (c.Amount - SUM(p.AmountPaid)) AS RemainingBalance
FROM Contract c
JOIN Vendor v ON c.VendorID = v.VendorID
JOIN Payments p ON c.ContractID = p.ContractID
GROUP BY c.ContractID, v.V_Name, c.Amount;
```



Query Query History

```
SELECT
    c.ContractID,
    v.V_Name,
    SUM(p.AmountPaid) AS TotalPaid,
    c.Amount AS ContractAmount,
    (c.Amount - SUM(p.AmountPaid)) AS RemainingBalance
FROM Contract c
JOIN Vendor v ON c.VendorID = v.VendorID
JOIN Payments p ON c.ContractID = p.ContractID
GROUP BY c.ContractID, v.V_Name, c.Amount;
```

Data Output Messages Notifications

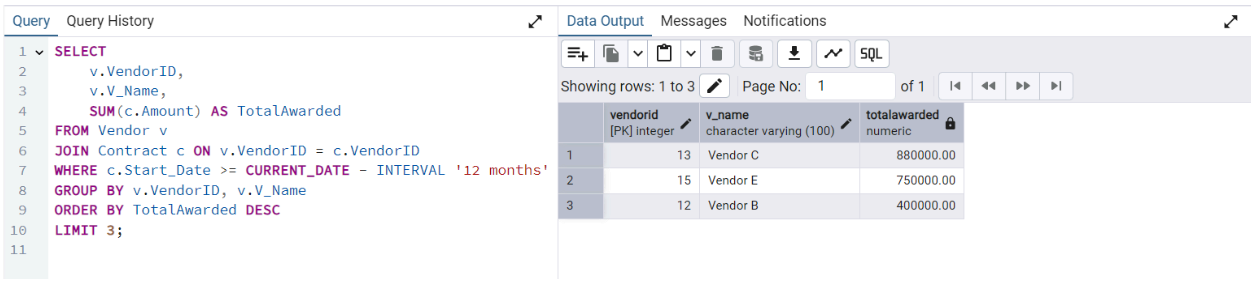
Showing rows: 1 to 1 Page No: 1 of 1

	contractid integer	v_name character varying (100)	totalpaid numeric	contractamount numeric (12,2)	remainingbalance numeric
1	21	Vendor C	500000.00	600000.00	100000.00

4. Top 3 Vendors by Total Contract Value (Past 12 Months)

```
SELECT
    v.VendorID,
    v.V_Name,
    SUM(c.Amount) AS TotalAwarded
FROM Vendor v
JOIN Contract c ON v.VendorID = c.VendorID
```

```
WHERE c.Start_Date >= CURRENT_DATE - INTERVAL '12 months'
GROUP BY v.VendorID, v.V_Name
ORDER BY TotalAwarded DESC
LIMIT 3;
```

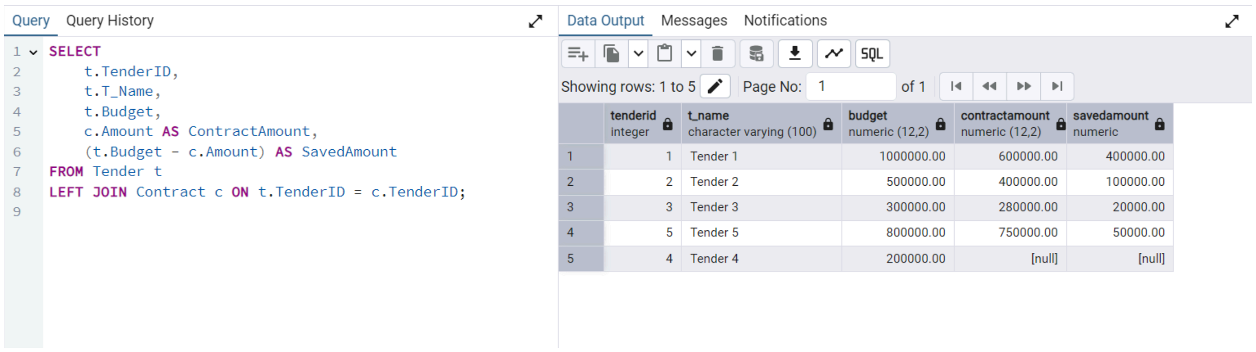


```
SELECT
    v.VendorID,
    v.V_Name,
    SUM(c.Amount) AS TotalAwarded
FROM Vendor v
JOIN Contract c ON v.VendorID = c.VendorID
WHERE c.Start_Date >= CURRENT_DATE - INTERVAL '12 months'
GROUP BY v.VendorID, v.V_Name
ORDER BY TotalAwarded DESC
LIMIT 3;
```

	vendorid [PK] integer	v_name character varying (100)	totalawarded numeric
1	13	Vendor C	880000.00
2	15	Vendor E	750000.00
3	12	Vendor B	400000.00

5. Total Budget and Expenditure Per Tender and Savings from Govt Side

```
SELECT
    t.TenderID,
    t.T_Name,
    t.Budget,
    c.Amount AS ContractAmount,
    (t.Budget - c.Amount) AS SavedAmount
FROM Tender t
LEFT JOIN Contract c ON t.TenderID = c.TenderID;
```



```
SELECT
    t.TenderID,
    t.T_Name,
    t.Budget,
    c.Amount AS ContractAmount,
    (t.Budget - c.Amount) AS SavedAmount
FROM Tender t
LEFT JOIN Contract c ON t.TenderID = c.TenderID;
```

	tenderid integer	t_name character varying (100)	budget numeric (12,2)	contractamount numeric (12,2)	savedamount numeric
1	1	Tender 1	1000000.00	600000.00	400000.00
2	2	Tender 2	500000.00	400000.00	100000.00
3	3	Tender 3	300000.00	280000.00	20000.00
4	5	Tender 5	800000.00	750000.00	50000.00
5	4	Tender 4	200000.00	[null]	[null]

<u>SOME OTHER QUERIES USEFUL:</u>

1. Audit Committee Member Evaluation

```
SELECT
    ac.UserID,
    ac.Name,
    ac.Role,
    COUNT(be.EvalID) AS Total_Evaluations,
    SUM(CASE WHEN be.ApprovalStatus = 'Approved' THEN 1 ELSE 0 END) AS
Approved_Count,
    SUM(CASE WHEN be.ApprovalStatus = 'Rejected' THEN 1 ELSE 0 END) AS
Rejected_Count
FROM AuditCommittee ac
LEFT JOIN BidEval be ON ac.UserID = be.UserID
GROUP BY ac.UserID, ac.Name, ac.Role
ORDER BY Total_Evaluations DESC;
```

2. List of All Tenders

```
SELECT * FROM Tender;
```

3. List of All Vendors

```
SELECT * FROM Vendor;
```

4. List of All Contracts Awarded to a Particular Vendor

```
SELECT * FROM Contract
WHERE VendorID = 3;
```

5. Lowest Bid for a Particular Tender

```
SELECT * FROM Bids
WHERE TenderID = 1
ORDER BY BidAmount ASC
LIMIT 1;
```

6. List of Bids Submitted by a Particular Vendor

```
SELECT * FROM Bids
WHERE VendorID = 3;
```

7. Tenders Still Open

```
SELECT * FROM Tender
WHERE Status = 'Open';
```

8. Upcoming Tender Deadlines

```
SELECT * FROM Tender
```

```
WHERE Deadline BETWEEN CURRENT_DATE AND (CURRENT_DATE + INTERVAL '30
days');
```

Future Aspiration And Scope Of Improvements-

1. Role-Based Access Control (RBAC) and User Authentication

- 🔐 Add a user authentication module to securely log in as admin, vendor, or auditor.
- 👥 Implement different permissions for each role (e.g., only auditors can evaluate bids).

2. Vendor Performance Analytics Dashboard

- 📊 Create analytics dashboards for:
  - Bid win rate
  - Contract fulfillment rate
  - Red-flag trends
- Use charts and graphs for admin insight.

3. Automatic Tender Notifications

- 📬 Email or SMS alert system:
  - New tender posted
  - Deadline reminders
  - Bid result notifications

—---x—--------------------------------------