**Normalizing the Relational Model and Creating a Normalized Oracle Database - Beta University Annual Fund**

**Step 6.1 - Begin with the relational schema that you developed at the end of Chapter 4 for the team project.** For each table on the list, identify functional dependencies and normalize the relation to BCNF. Then decide whether the resulting tables should be implemented in that form. If not, explain why.

**Potential Donor**(donorId,firstName, lastName, street, city, state, zip, country, countryCode, areaCode, telNumber,donorCircle, spouseName, amountDonatedLastYear, amountPledgedThisYear, amountPaidThisYear, *volunteerfirstName, volunteerLastName*)

**MatchingCorp**(matchCorpName,street, city, state, zip, country, countryCode, areaCode, telNumber))

**Event**(eventName, eventDate, eventTime, eventLocation, eventOrganizer, eventTotalPledged eventTotalPaid)

**ClassRep**(graduationYear, firstName, lastName, street, city, state, zip, country,  countryCode, areaCode, telNumber)

**Volunteer**(firstName, lastName, street, city, state, zip, country, countryCode, areaCode, telNumber)

**Pledge**(pledgeNumber,  pledgeAmount, pledgeDate, numPaymentsChosen, pledgeAmountPaid, numPaymentsMade, *donorId, eventName*)

**Payment**(*pledgeNumber*, datePaid, amount Paid, paymentMethod, creditCardType, credit Card

**PotentialDonor-Event**(*donorId, eventName*)

**Pledge-MatchingCorp**(*pledgeNumber, matchCorpName,*spouse)

**Donor-GradYear**(*donorId*, graduationYear)

**Donor-Categories**(*donorId*, donorCategory)

1. **Potential Donor:**

**Potential Donor**(donorId,firstName, lastName, street, city, state, zip, country, countryCode, areaCode, telNumber,donorCircle, spouseName, amountDonatedLastYear, amountPledgedThisYear, amountPaidThisYear, *volunteerfirstName, volunteerLastName*)

**Functional Dependencies:**

**donorId** → firstName, lastName, street, city, state, zip, country, countryCode, areaCode, telNumber, donorCircle, spouseName, amountDonatedLastYear, amountPledgedThisYear, amountPaidThisYear

**donorId** → volunteerfirstName, volunteerLastName

**Normalization to BCNF:**

To normalize the relation to BCNF , I added **volunteerID** to the **PotentialDonor** table to remove the partial dependency:

**PotentialDonor**(donorId, firstName, lastName, street, city, state, zip, country, countryCode, areaCode, telNumber, donorCircle, spouseName, amountDonatedLastYear, amountPledgedThisYear, amountPaidThisYear, volunteerID)

**Decision:**

Now, this schema is already in BCNF as there are no partial or transitive dependencies.

1. **MatchingCorp Table:**

MatchingCorp(matchCorpName, street, city, state, zip, country, countryCode, areaCode, telNumber)

**Functional Dependencies:**

**matchCorpName →** street, city, state, zip, country, countryCode, areaCode, telNumber

**Decision:**

This is already in BCNF since the matchCorpName functionally determines all the other attributes. No further normalization is needed.

1. **Event Table:**

Event(eventName, eventDate, eventTime, eventLocation, eventOrganizer, eventTotalPledged, eventTotalPaid)

**Functional Dependencies:**

**eventName →** eventDate, eventTime, eventLocation, eventOrganizer, eventTotalPledged, eventTotalPaid

**Decision:**

This is already in BCNF because eventName determines all other attributes.

1. **ClassRep Table:**

ClassRep(graduationYear, firstName, lastName, street, city, state, zip, country, countryCode, areaCode, telNumber)

**Functional Dependencies:**

**(graduationYear, firstName, lastName)** **→** street, city, state, zip, country, countryCode, areaCode, telNumber

**Decision:**

This is already in BCNF because the combination of graduationYear, firstName, and lastName determines all other attributes.

1. **Volunteer Table:**

Volunteer(volunteerID, firstName, lastName, street, city, state, zip, country, countryCode, areaCode, telNumber)

**Functional Dependencies:**

**(volunteerID) →** firstName, lastName, street, city, state, zip, country, countryCode, areaCode, telNumber

**Decision:**

This is already in BCNF because **volunteerID** determines all other attributes.

1. **Pledge Table:**

Pledge(pledgeNumber, pledgeAmount, pledgeDate, numPaymentsChosen, pledgeAmountPaid, numPaymentsMade, donorId, eventName)

**Functional Dependencies:**

**pledgeNumber →** pledgeAmount, pledgeDate, numPaymentsChosen, pledgeAmountPaid, numPaymentsMade, donorId, eventName

**Decision:**

This is already in BCNF because pledgeNumber determines all other attributes.

1. **Payment Table:**

Pledge(*pledgeNumber*, datePaid, amount Paid, paymentMethod, creditCardType, credit Card Number, *matchCorpName*)

**Functional Dependencies:**

**(*pledgeNumber*, datePaid) →** amount Paid, paymentMethod, creditCardType, credit Card Number, *matchCorpName*

**Decision:**

This is already in BCNF because the combination of ***pledgeNumber*, datePaid** determines all other attributes.

**Normalized Schema:   
Potential Donor**(donorId,firstName, lastName, street, city, state, zip, country, countryCode, areaCode, telNumber,donorCircle, spouseName, amountDonatedLastYear, amountPledgedThisYear, amountPaidThisYear, *volunteerID*)

**MatchingCorp**(matchCorpName,street, city, state, zip, country, countryCode, areaCode, telNumber))

**Event**(eventName, eventDate, eventTime, eventLocation, eventOrganizer, eventTotalPledged eventTotalPaid)

**ClassRep**(graduationYear, firstName, lastName, street, city, state, zip, country,  countryCode, areaCode, telNumber)

**Volunteer**(volunteerID, firstName, lastName, street, city, state, zip, country, countryCode, areaCode, telNumber)

**Pledge**(pledgeNumber,  pledgeAmount, pledgeDate, numPaymentsChosen, pledgeAmountPaid, numPaymentsMade, *donorId, eventName*)

**Payment**(*pledgeNumber*, datePaid, amount Paid, paymentMethod, creditCardType, credit Card Number, *matchCorpName)*

**PotentialDonor-Event**(*donorId, eventName*)

**Pledge-MatchingCorp**(*pledgeNumber, matchCorpName,*spouse)

**Donor-GradYear**(*donorId*, graduationYear)

**Donor-Categories**(donorId, donorCategory)

**Step 6.2 – Review and update the data dictionary (as needed).**

Review the data dictionary that was created in the Chapter 5 Assignment. Update the data dictionary (as needed). If there are no updates, then relist your Chapter 5 data dictionary.

|  |  |
| --- | --- |
| classRepAddress | The mailing address of class representative. |
| classRepDonationAmount | The total donation amount collected from the class representative. |
| classRepID | The unique ID for class representative. |
| classRepLastYearDonationAmount | The total donation amount collected by the class representative in the previous year. |
| classRepName | The name of class representative. |
| classRepPhoneNumber | The phone number of class representative. |
| corporationCity | The city of mailing address of a corporation. |
| corporationCountry | The country of mailing address of a corporation |
| corporationID | The unique ID of a corporation. |
| corporationName | The name of a corporation. |
| corporationState | The state of mailing address of a corporation. |
| corporationStreet | The street and building number of mailing address of a corporation. |
| corporationZipCode | The postal zip code of a corporation |
| currentYearPledgeAmount | The total amount pledged by the donor for the current year. |
| donationAmount | The total amount of money contributed as a donation. |
| donationAmountByPhonothon | The total amount of donation collected through phonathon fundraising efforts. |
| donorAddressFromMultipleCategories | The mailing address of a donor who belongs to multiple donor categories. |
| donorAddressWithMultipleGradYears | The mailing address of a donor who has graduated in multiple years. (A donor may have more than one degree from the university, so he or she may have more than one graduation year) |
| donorCardCVV | The Card Verification Value (CVV) of the donor's credit or debit card, used for security purposes during transactions. |
| donorCardExpirationDate | The card expiration date of donor’s credit or debit card. |
| donorCategory | The category of a donor. |
| donorCity | The city of mailing address of a donor. |
| donorCircle | The classification of donor based on the size of their contributions. |
| donorCountry | The country of mailing address of a donor. |
| donorCreditCardNumber | The credit card number of a donor. |
| donorEmailAddress | The email address of a donor. |
| donorEnclosedAmount | The total amount of money included with a donation sent by the donor. |
| donorFirstName | The first name of a donor. |
| donorGraduationYear | The year in which the donor graduated. |
| donorIDFromMultipleCategories | The unique ID assigned to a donor who belongs to multiple donor categories. |
| donorIDWithMultipleGradYears | The unique ID assigned to a donor who has graduated in multiple years. |
| donorLastName | The last name of a donor. |
| donorNameFromMultipleCategories | The name of a donor who belongs to multiple donor categories. |
| donorNameWithMultipleGradYears | The name of a donor who has graduated in multiple years. |
| donorNumberOfPayments | The total number of payments made by the donor towards their pledged amount or donation. |
| donorPaymentMethod | The method used by the donor to make a payment. |
| donorPhoneNumber | The phone number of a donor. |
| donorPhoneFromMultipleCategories | The phone number of a donor who belongs to multiple donor categories. |
| donorPhoneWithMultipleGradYears | The phone number of donor who has graduated in multiple years. |
| donorPledgeGiftDate | The date on which the donor made their pledge or commitment to contribute a gift. |
| donorPledgedAmount | The total amount that the donor has committed to contribute. |
| donorPreviousPaymentDate | The previous payment date of a donor. |
| donorState | The state of mailing address of a donor. |
| donorStreet | The street of mailing address of a donor. |
| donorZipCode | The postal zip code of mailing address of a donor. |
| donorID | The unique ID assigned to a donor. |
| eventID | The unique ID assigned to an event. |
| eventOrganizedDate | The date on which the event was organized. |
| eventType | The type of event. |
| giftID | The unique ID assigned to a gift. |
| giftReceivedDate | The date on which the gift was received. |
| installmentPaymentDonorID | The unique ID assigned to the donor associated with an installment payment. |
| installmentPaymentID | The unique ID of a payment that has been received in installments. |
| lastYearPledgeAmount | The total amount pledged by the donor in the previous year. |
| missedPaymentDonorID | The unique ID assigned to the donor who has missed a scheduled payment. |
| paymentDueDate | The due date of a payment. |
| paymentID | The unique ID assigned to a payment. |
| paymentReceivedDate | The date on which a payment was received. |
| pledgeAmount | The total amount that a donor has promised to contribute. |
| pledgeReceivedDate | The date on which the donor's pledge was recorded. |
| pledgingDate | The date on which the donor made their pledge to contribute. |
| potentialDonorAddress | The mailing address of a potential donor. |
| potentialDonorID | The unique ID assigned to a potential donor. |
| potentialDonorName | The name of a potential donor. |
| potentialDonorPhoneNumber | The phone number of a potential donor. |
| spouseCorporationAddress | The mailing address of the corporation of donor's spouse. |
| spouseCorporationName | The name of a corporation of a donor’s spouse. |
| spouseName | The name of a donor’s spouse. |
| volunteerAddress | The mailing address of a volunteer. |
| volunteerEmailAddress | The email address of a volunteer. |
| volunteerID | The unique ID assigned to a volunteer. |
| volunteerName | The name of a volunteer. |
| volunteerPhoneNumber | The phone number of a volunteer. |

**Step 6.3 – Review and update the list of assumptions (as needed).**

Review the list of assumptions that was created in the Chapter 5 Assignment. Update the list of assumptions (as needed). If there are no updates, then relist your Chapter 5 list of assumptions.

* Names of donors are not unique, but names of volunteers and matching corporations are unique.
* A person may belong to more than one category. For example, a parent may also be an alumni.
* A person may have more than one degree from the university, so he or she may have more than one graduation year. That person’s gift is counted in the totals and percentages for all his or her graduation years.
* There is exactly one class representative for each graduation year.
* A potential donor receives a call from at most one class representative or volunteer each fund year.
* A donor may give more than one pledge during a fund year.
* A person may attend several fundraising events within a year.
* It is desirable to know which pledges are due to a fundraiser, but not necessary to know which are due to a class coordinator or volunteer’s call.
* All pledges for a given fund year are collected within that year.
* Matching gifts are paid by corporations in a single payment within the same fund year as the donor’s gift.
* At the end of each fund year, the entire database is archived, and a new database for the new fund year is created. The database name identifies the year.
* All addresses consist of street, city, state, zip code or postal code, and country.
* All telephone numbers consist of country code, area code, and number.

**Step 6.4 - For each table, write out the table name and the attribute names, data types, and data lengths and all constraints, using the conventions of the Oracle Database that you will use for implementation of a normalized database for the team project.**

**Answer:**

**PotentialDonor:**

TABLE PotentialDonor

donorID VARCHAR2(20) PRIMARY KEY,

firstName VARCHAR2(15) NOT NULL UNIQUE,

lastName VARCHAR2(15) NOT NULL UNIQUE,

street VARCHAR2(50) NOT NULL,

city VARCHAR2(15) NOT NULL,

state CHAR(2) NOT NULL,

zip CHAR(5) NOT NULL CHECK (LENGTH(zip) = 5),

country VARCHAR2(30),

countryCode VARCHAR2(3),

areaCode CHAR(3),

telNumber CHAR(7),

donorCircle VARCHAR2(20),

spouseName VARCHAR2(30),

amountDonatedLastYear NUMBER(10, 2) DEFAULT 0.00,

amountPledgedThisYear NUMBER(10, 2) DEFAULT 0.00,

amountPaidThisYear NUMBER(10, 2) DEFAULT 0.00,

volunteerID VARCHAR2(20),

FOREIGN KEY (volunteerID) REFERENCES Volunteer(volunteerID)

**MatchingCorp:**

TABLE MatchingCorp

matchCorpName VARCHAR2 20 PRIMARY KEY

street VARCHAR2 50 NOT NULL

city VARCHAR2 15 NOT NULL

state CHAR 2 NOT NULL

zip CHAR 5 NOT NULL CHECK (LENGTH(zip) = 5)

country VARCHAR2 30

countryCode VARCHAR2 3

areaCode CHAR 3

telNumber CHAR 7

**Event:**

TABLE Event

eventName VARCHAR2 20 PRIMARY KEY

eventDate DATE NOT NULL

eventTime TIMESTAMP NOT NULL

eventLocation VARCHAR2 50 NOT NULL

eventOrganizer VARCHAR2 30

eventTotalPledged NUMBER 10,2 DEFAULT 0.00

eventTotalPaid NUMBER 10,2 DEFAULT 0.00

**ClassRep:**

TABLE ClassRep

graduationYear NUMBER 4 PRIMARY KEY

firstName VARCHAR2 15 NOT NULL UNIQUE

lastName VARCHAR2 15 NOT NULL UNIQUE

street VARCHAR2 50 NOT NULL

city VARCHAR2 15 NOT NULL

state CHAR 2 NOT NULL

zip CHAR 5 NOT NULL CHECK (LENGTH(zip) = 5)

country VARCHAR2 30

countryCode VARCHAR2 3

areaCode CHAR 3

telNumber CHAR 7

**Volunteer:**

TABLE Volunteer

volunteerID VARCHAR2(20) PRIMARY KEY,

firstName VARCHAR2(15) NOT NULL,

lastName VARCHAR2(15) NOT NULL,

street VARCHAR2(50) NOT NULL,

city VARCHAR2(15) NOT NULL,

state CHAR(2) NOT NULL,

zip CHAR(5) NOT NULL CHECK (LENGTH(zip) = 5),

country VARCHAR2(30),

countryCode VARCHAR2(3),

areaCode CHAR(3),

telNumber CHAR(7)

**Pledge:**

TABLE Pledge

pledgeNumber VARCHAR2 15 PRIMARY KEY

pledgeAmount NUMBER 10,2 DEFAULT 0.00

pledgeDate DATE NOT NULL

numPaymentsChosen NUMBER 3 NOT NULL

pledgeAmountPaid NUMBER 10,2 DEFAULT 0.00

numPaymentsMade NUMBER 3 NOT NULL

donorId VARCHAR2 20 FOREIGN KEY

eventName VARCHAR2 20 FOREIGN KEY

**Payment:**

TABLE Payment

pledgeNumber VARCHAR2 15 FOREIGN KEY

datePaid Date PRIMARY KEY

amountPaid NUMBER 10,2 DEFAULT 0.00

paymentMethod VARCHAR2 20 NOT NULL

creditCardType VARCHAR2 15 NOT NULL

creditCardNumber NUMBER 20 NOT NULL

matchCorpName VARCHAR2 20 FOREIGN KEY

**Step 6.5 – Step 6.8 - Creating a normalized database for Team Project.**

For each step:

- Use your Team’s Oracle User ID to run the SQL statements in the database. (Do not use SYS or SYSTEM).

- Write out the SQL statements that will be executed in the database (in addition to showing the SQL Statements in the screenshot).

- Execute the SQL statements in the database. Show your work by providing screenshots that include the current user id, current system date, executing the SQL statements in the database, and the results. Any screenshots that do not include this information will receive no credit. To display the current user and current date, run the following statement before running each SQL statement:

SQL> select user||' '||to\_char(sysdate, 'DD-MON-YYYY HH24:MI:SS') from dual;

**Step 6.5 - Design SQL statements to create all objects (tables, constraints, indexes) needed to implement the design. Then execute the SQL statements in the database.**

**PotentialDonor:**

CREATE TABLE PotentialDonor (

donorID VARCHAR2(20) PRIMARY KEY,

firstName VARCHAR2(15) NOT NULL UNIQUE,

lastName VARCHAR2(15) NOT NULL UNIQUE,

street VARCHAR2(50) NOT NULL,

city VARCHAR2(15) NOT NULL,

state CHAR(2) NOT NULL,

zip CHAR(5) NOT NULL CHECK (LENGTH(zip) = 5),

country VARCHAR2(30),

countryCode VARCHAR2(3),

areaCode CHAR(3),

telNumber CHAR(7),

donorCircle VARCHAR2(20),

spouseName VARCHAR2(30),

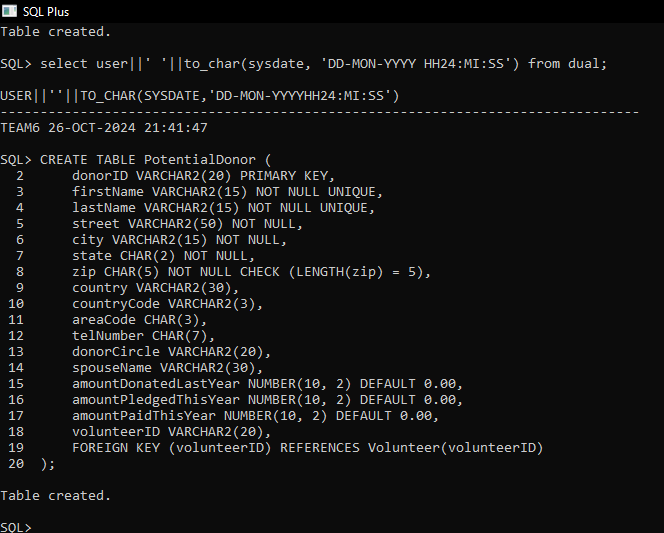
amountDonatedLastYear NUMBER(10, 2) DEFAULT 0.00,

amountPledgedThisYear NUMBER(10, 2) DEFAULT 0.00,

amountPaidThisYear NUMBER(10, 2) DEFAULT 0.00,

volunteerID VARCHAR2(20)

);



**MatchingCorp:**

CREATE TABLE MatchingCorp (

matchCorpName VARCHAR2(20) PRIMARY KEY,

street VARCHAR2(50) NOT NULL,

city VARCHAR2(15) NOT NULL,

state CHAR(2) NOT NULL,

zip CHAR(5) NOT NULL CHECK (LENGTH(zip) = 5),

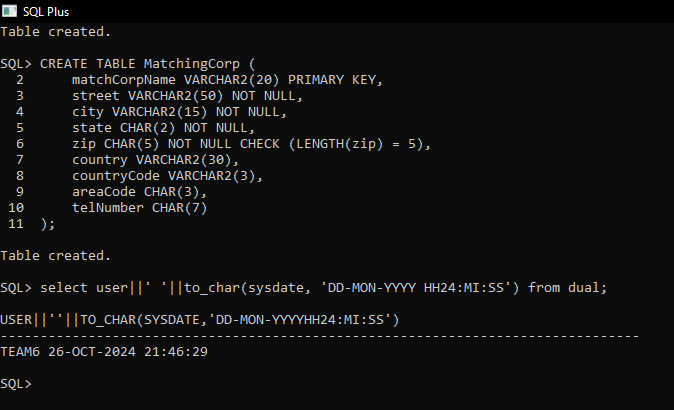
country VARCHAR2(30),

countryCode VARCHAR2(3),

areaCode CHAR(3),

telNumber CHAR(7)

);



**Event:**

CREATE TABLE Event (

eventName VARCHAR2(20) PRIMARY KEY,

eventDate DATE NOT NULL,

eventTime TIMESTAMP NOT NULL,

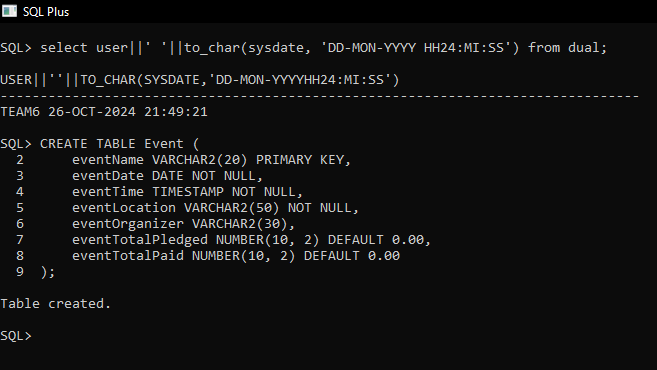
eventLocation VARCHAR2(50) NOT NULL,

eventOrganizer VARCHAR2(30),

eventTotalPledged NUMBER(10, 2) DEFAULT 0.00,

eventTotalPaid NUMBER(10, 2) DEFAULT 0.00

);



**ClassRep:**

CREATE TABLE ClassRep (

graduationYear NUMBER(4) PRIMARY KEY,

firstName VARCHAR2(15) NOT NULL UNIQUE,

lastName VARCHAR2(15) NOT NULL UNIQUE,

street VARCHAR2(50) NOT NULL,

city VARCHAR2(15) NOT NULL,

state CHAR(2) NOT NULL,

zip CHAR(5) NOT NULL CHECK (LENGTH(zip) = 5),

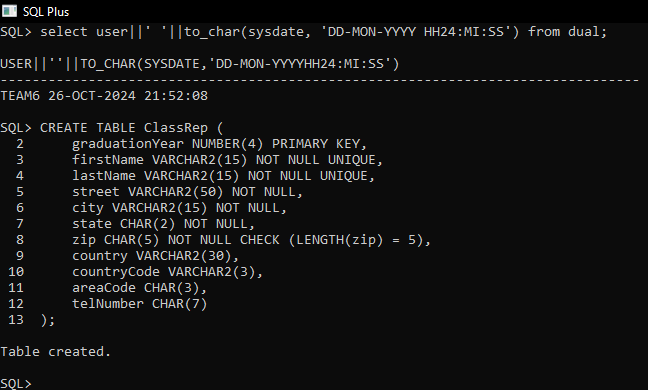
country VARCHAR2(30),

countryCode VARCHAR2(3),

areaCode CHAR(3),

telNumber CHAR(7)

);



**Volunteer:**

CREATE TABLE Volunteer (

volunteerID VARCHAR2(20) PRIMARY KEY,

firstName VARCHAR2(15) NOT NULL,

lastName VARCHAR2(15) NOT NULL,

street VARCHAR2(50) NOT NULL,

city VARCHAR2(15) NOT NULL,

state CHAR(2) NOT NULL,

zip CHAR(5) NOT NULL CHECK (LENGTH(zip) = 5),

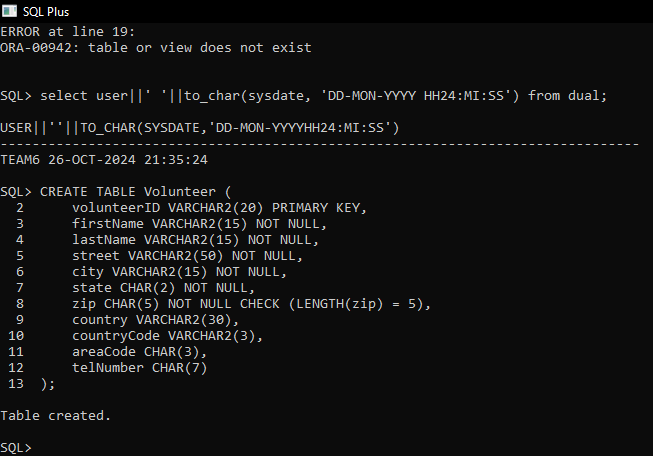
country VARCHAR2(30),

countryCode VARCHAR2(3),

areaCode CHAR(3),

telNumber CHAR(7)

);



**Pledge:**

CREATE TABLE Pledge (

pledgeNumber VARCHAR2(15) PRIMARY KEY,

pledgeAmount NUMBER(10, 2) DEFAULT 0.00,

pledgeDate DATE NOT NULL,

numPaymentsChosen NUMBER(3) NOT NULL,

pledgeAmountPaid NUMBER(10, 2) DEFAULT 0.00,

numPaymentsMade NUMBER(3) NOT NULL,

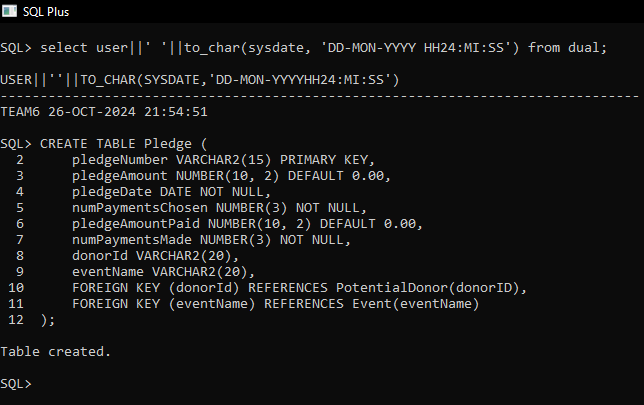
donorId VARCHAR2(20),

eventName VARCHAR2(20),

FOREIGN KEY (donorId) REFERENCES PotentialDonor(donorID),

FOREIGN KEY (eventName) REFERENCES Event(eventName)

);



**Payment:**

CREATE TABLE Payment (

pledgeNumber VARCHAR2(15),

datePaid DATE PRIMARY KEY,

amountPaid NUMBER(10, 2) DEFAULT 0.00,

paymentMethod VARCHAR2(20) NOT NULL,

creditCardType VARCHAR2(15) NOT NULL,

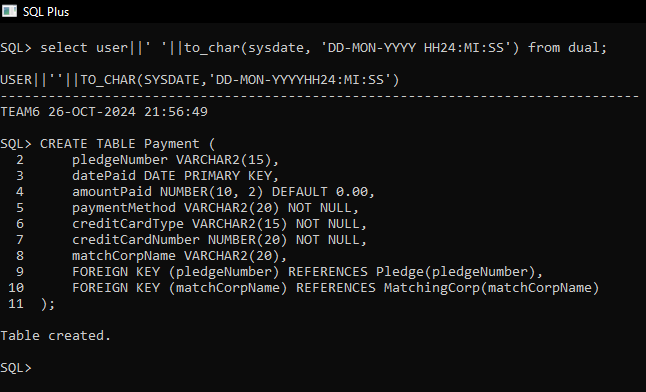
creditCardNumber NUMBER(20) NOT NULL,

matchCorpName VARCHAR2(20),

FOREIGN KEY (pledgeNumber) REFERENCES Pledge(pledgeNumber),

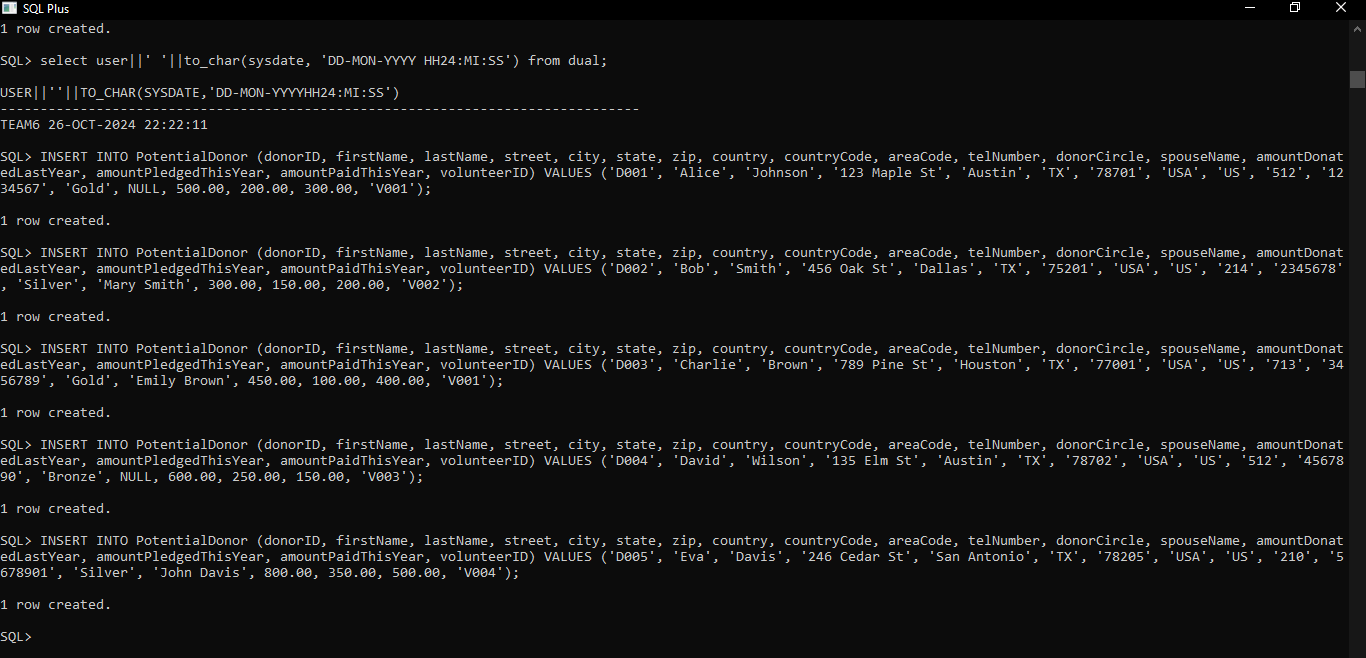
FOREIGN KEY (matchCorpName) REFERENCES MatchingCorp(matchCorpName)

);

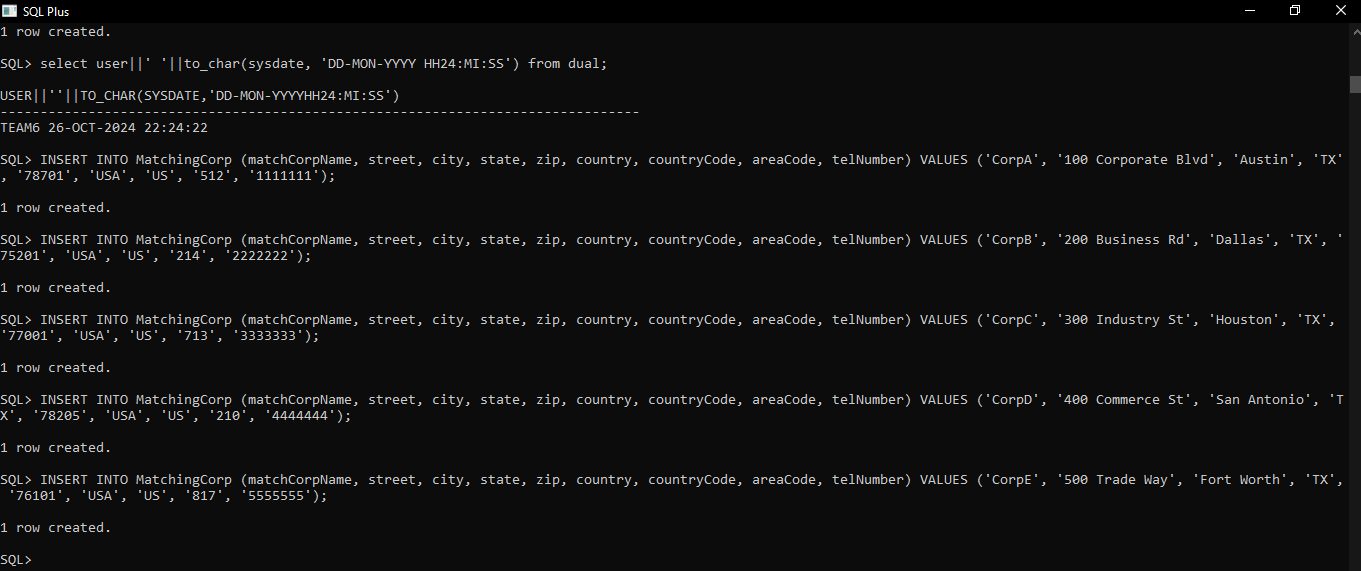


**Step 6.6 - Design SQL statements to insert at least five records in each table, preserving all constraints.** **Then execute the SQL statements to insert the records into the tables.**

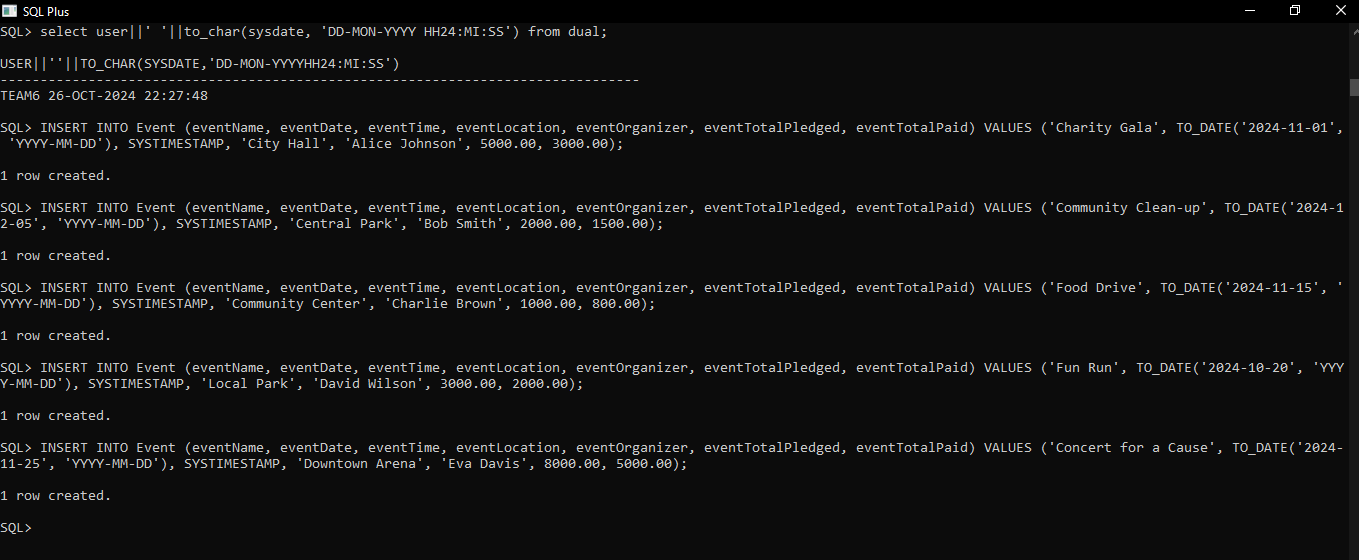
**PotentialDonor:**



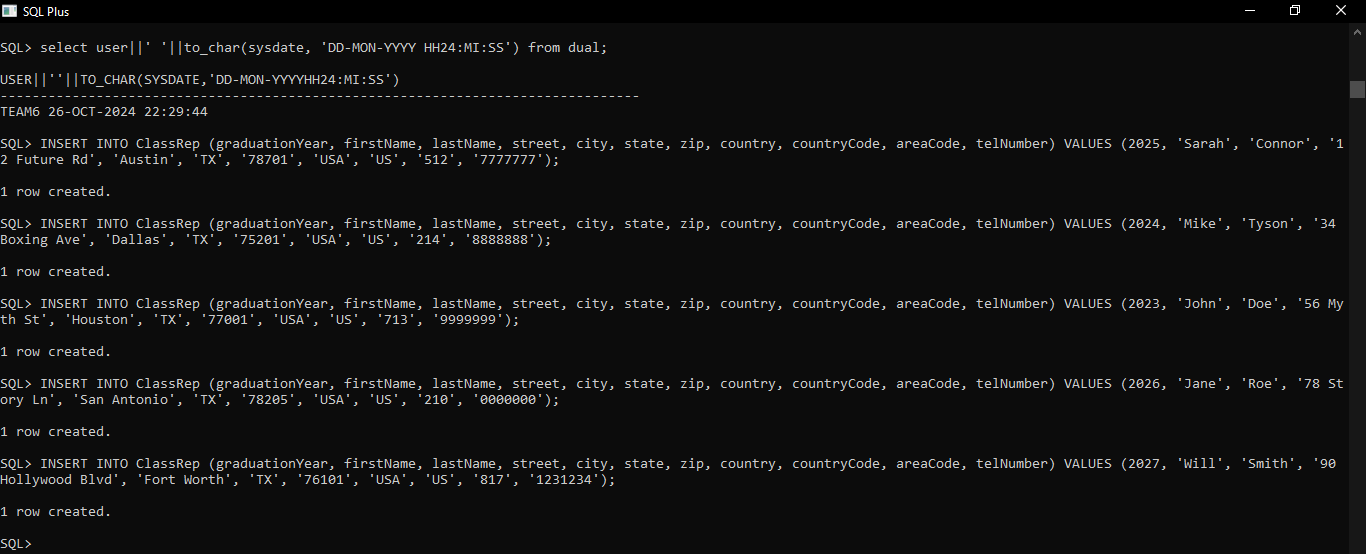
**MatchingCorp:**



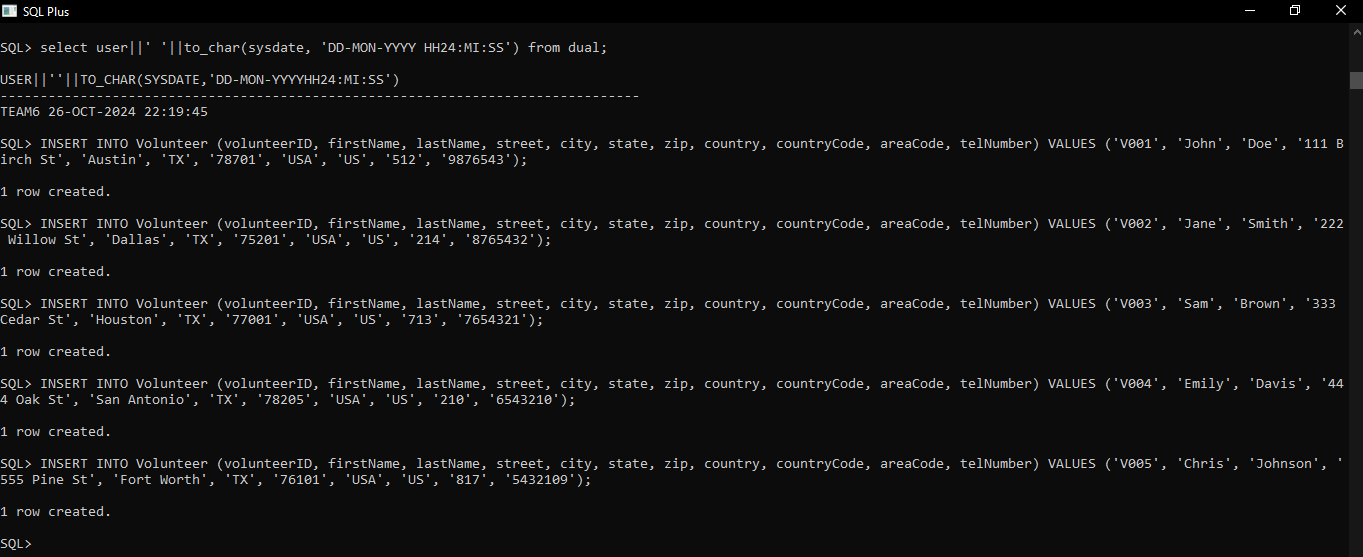
**Event:**



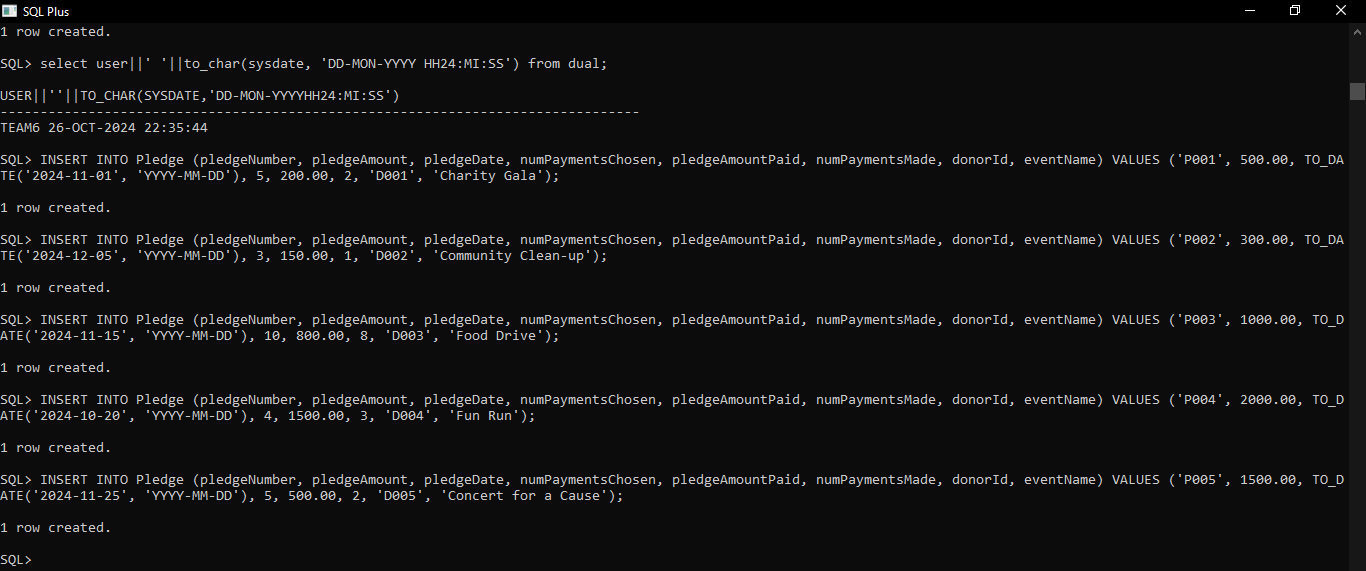
**ClassRep:**



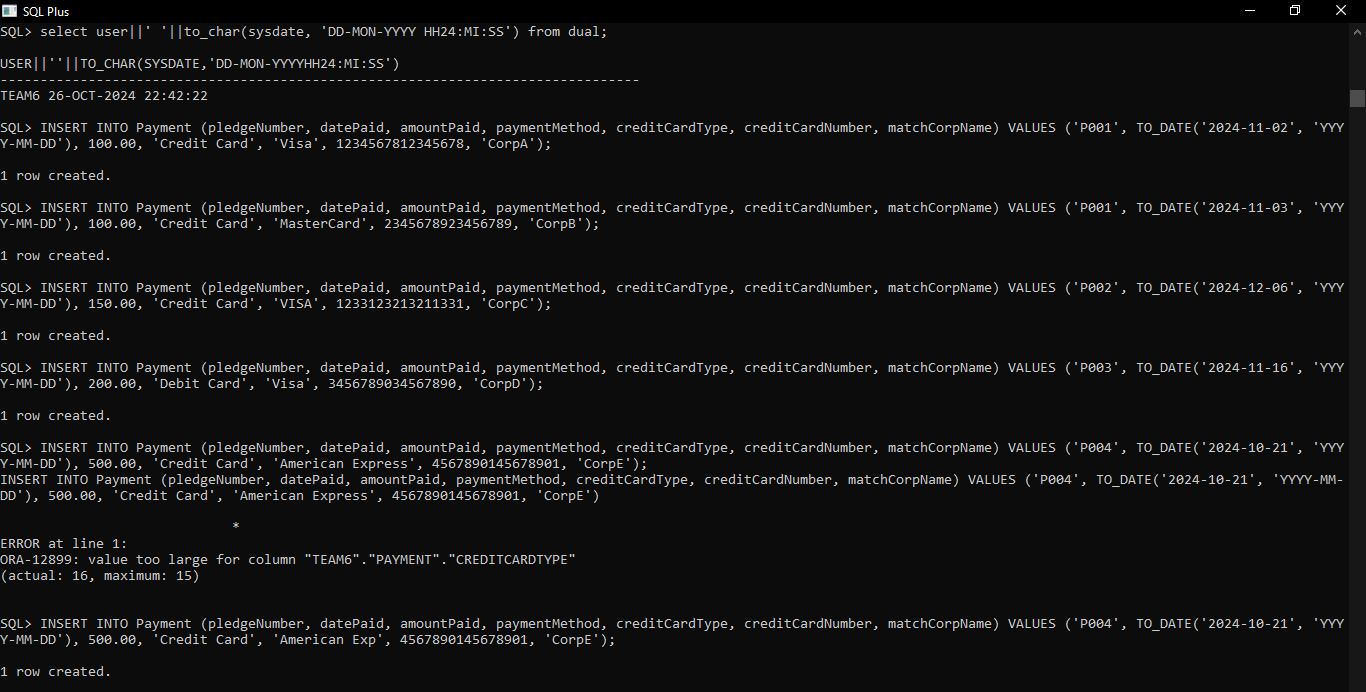
**Volunteer:**



**Pledge:**



**Payment:**



**Step 6.7 - Design five non-routine questions that will request information from the database for the team project. Design a SQL Statement for each question that will be executed in the Oracle database. Then execute each SQL statement in the Oracle Database.**

These five non-routine requests should be different from the ones you created in Chapter 5.

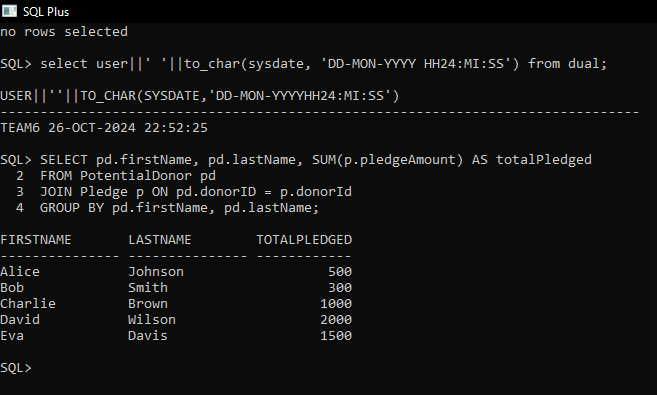
**Question 1: What are the details of donors who have participated in events, including their total pledged amount?**

SELECT pd.firstName, pd.lastName, SUM(p.pledgeAmount) AS totalPledged

FROM PotentialDonor pd

JOIN Pledge p ON pd.donorID = p.donorId

GROUP BY pd.firstName, pd.lastName;



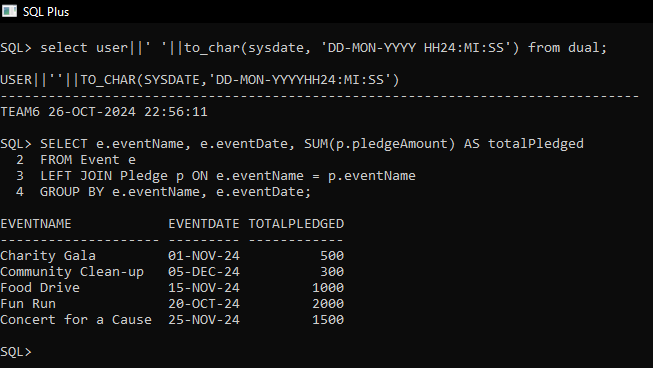
**Question 2:** **What are the details of all events and the total amount pledged for each event?**

SELECT e.eventName, e.eventDate, SUM(p.pledgeAmount) AS totalPledged

FROM Event e

LEFT JOIN Pledge p ON e.eventName = p.eventName

GROUP BY e.eventName, e.eventDate;

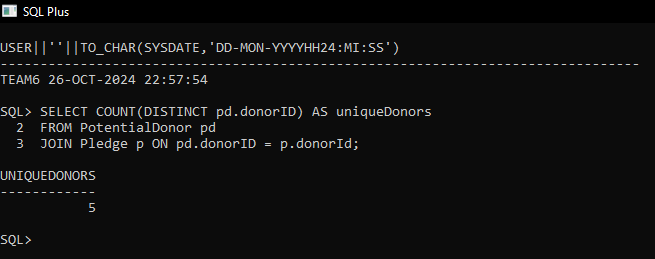


**Question 3: How many unique donors have made pledges to the events?**

SELECT COUNT(DISTINCT pd.donorID) AS uniqueDonors

FROM PotentialDonor pd

JOIN Pledge p ON pd.donorID = p.donorId;



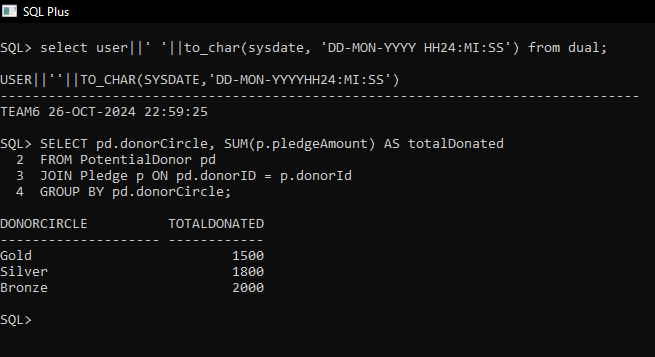
**Question 4: What is the total amount donated by donors grouped by their donor circle?**

SELECT pd.donorCircle, SUM(p.pledgeAmount) AS totalDonated

FROM PotentialDonor pd

JOIN Pledge p ON pd.donorID = p.donorId

GROUP BY pd.donorCircle;



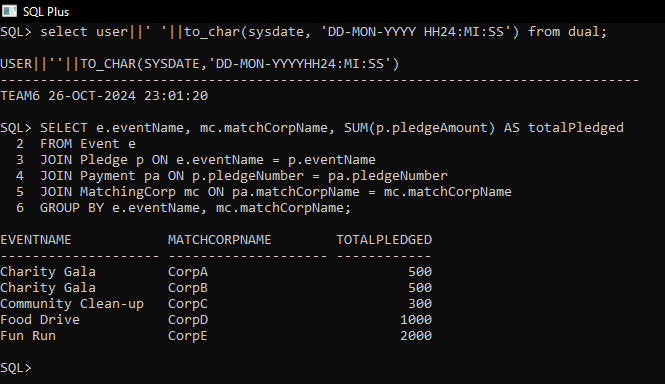
**Question 5: Which matching corporations have contributed to events, and what is the total amount they have paid for those events?**

SELECT mc.matchCorpName, SUM(p.amountPaid) AS totalPaid

FROM MatchingCorp mc

JOIN Payment p ON mc.matchCorpName = p.matchCorpName

GROUP BY mc.matchCorpName;



**Step 6.8 - Design one trigger for your project. Then create the trigger in the database. Then design and execute SQL statements to demonstrate that the trigger is working as expected.** Create a trigger that will update an attribute on one table whenever a row is updated on another table. To demonstrate that the trigger is working as expected, provide a screenshot of the data before and after the trigger is executed. This trigger should be different from the one you created in Chapter 5.

CREATE OR REPLACE TRIGGER update\_classrep\_city

AFTER UPDATE OF city ON Volunteer

FOR EACH ROW

BEGIN

UPDATE ClassRep

SET city = :NEW.city

WHERE firstName = :OLD.firstName AND lastName = :OLD.lastName;

END;

/

