University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: October 14, 2024

Group Number: 116

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Quang Duy Do	49549538	n6k8c	jackydo1974@gmail.com
Hai Son Vu	23411960	f0f5u	vuhaison16@gmail.com
Sabir Shaikh	65129090	c0d1a	mohammedsabirshaikh2@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

University of British Columbia, Vancouver
Department of Computer Science

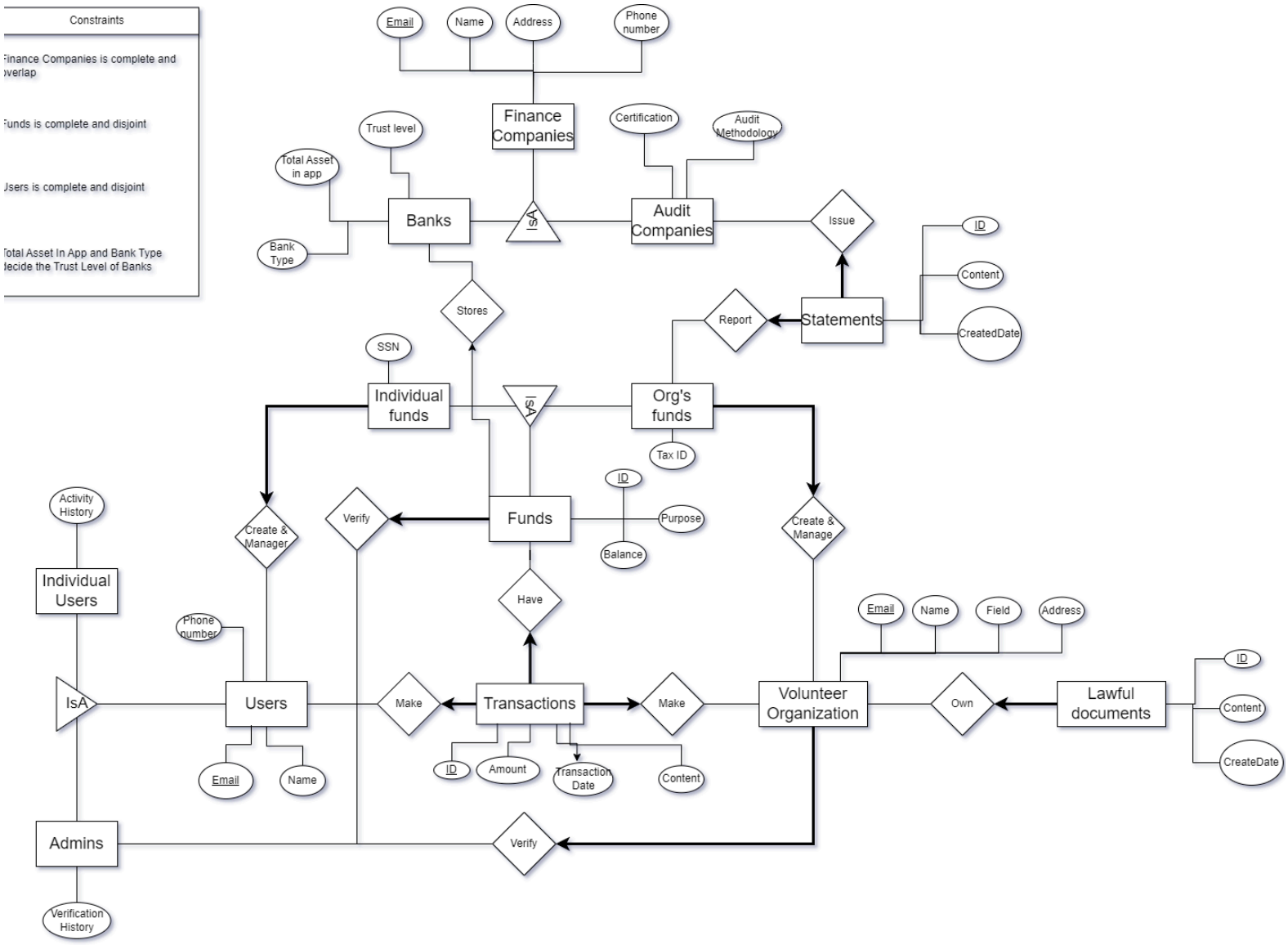
---

1. **Brief project summary** (~2-3 sentences):
    - This will help TAs remember details about your project, especially when they are managing multiple projects.

- A Charity Donation management database, where the best securities are applied for organizations to conduct collecting money and creating funds (which requires verification from Admin and statements from Financial Companies).
- Users and organizations can donate money to funds.
- Finance Companies are the storage of the money, and providing statements about the use of money
- Organizations need to provide legally lawful documents and verification from admins in order to start and open a Fund.

2. **ER diagram** (from item #3 below):

- We changed the mail to email in Users, Finance Companies and Organizations
- We changed the participation and relationship between transaction and funds - from many-many into one-many
- We added a Trust Level attribute to Banks for risk assessment in the future.
- TAs have commented about our primary keys as we chose all IDs which are artificial keys as primary keys. As a result, I chose Email for users, finance companies and organizations as primary keys. The transactions and funds must still have ID as primary key since they don't have anything unique about them.
- We added the constraints about our ISA hierarchy.
- We added a Trust Level for Banks. Trust Level is calculated based on Total Asset In Application and Bank Type.
- We changed the names of the Date attribute of LawfulDocument and Transaction into CreatedDate; and Date of Transaction into TransactionDate to avoid the same reserved data type with the database management tool.



4. **Schema derived from the ER diagram**:

- Follow instructions from your lectures for translating the ER diagram to the relational model.
- For each table:
  - List the table definition (e.g., `Table1(attr1: domain1, attr2: domain2, ...)`). Be sure to include domains for each attribute.

- Specify the **primary key (PK)**, **candidate key (CK)**, **foreign keys (FK)**, and any other constraints (e.g., **NOT NULL**, **UNIQUE**, etc.).

Users(Email: VARCHAR(255), Name: VARCHAR(50), PhoneNumber: CHAR(10))

Users

Users (

Email: VARCHAR(255)

Name: VARCHAR(50)

PhoneNumber: CHAR(10)

)

Primary Key (PK): Email

Candidate Key (CK): Email

Constraints:

- NOT NULL (Email, Name, PhoneNumber)

- UNIQUE (Email)

IndividualUser

IndividualUser (

Email: VARCHAR(255)

ActivityHistory: VARCHAR(255)

)

Primary Key (PK): Email

Candidate Key (CK): Email

Foreign Key (FK): UserEmail (Email references Users)

Constraints:

- NOT NULL (Email, ActivityHistory)

- UNIQUE (Email)

Admin

Admin (

Email: VARCHAR(255)

VerificationHistory: VARCHAR(255)

)

Primary Key (PK): Email

Candidate Key (CK): Email

Foreign Key (FK): UserEmail (Email references Users)

Constraints:

- NOT NULL (Email, VerificationHistory)

- UNIQUE (Email)

VolunteerOrganization

VolunteerOrganization (

Email: VARCHAR(255)

Name: VARCHAR(100)

Field: VARCHAR(50)

Address: VARCHAR(255)

Verification: VARCHAR(255)

)

Primary Key (PK): Email

Candidate Key (CK): (Name, Field)

Foreign Key (FK): Verification (Verification references Admin)

Constraints:

- NOT NULL (Email, Name, Field, Address, Verification)

- UNIQUE (Email)

Transaction

Transaction (

ID: INT

Amount: DECIMAL(10, 2)

TransactionDate: DATE

Content: VARCHAR(255)

UserEmail: VARCHAR(255)

OrganizationEmail: VARCHAR(255)

FundID: INTEGER

)

Primary Key (PK): ID

Candidate Key (CK): ID

Foreign Key (FK): ['UserEmail (UserEmail references Users)', 'OrganizationEmail (OrganizationEmail references VolunteerOrganization)', 'FundID (FundID references Fund)']

Constraints:

- NOT NULL (ID, Amount, TransactionDate, Content, UserEmail, OrganizationEmail, FundID)

- UNIQUE (ID)

LawfulDocuments

LawfulDocuments (

ID: INTEGER

Content: VARCHAR(255)

CreatedDate: DATE

OwnerOrganizationEmail: VARCHAR(255)

)

Primary Key (PK): ID

Candidate Key (CK): ID

Foreign Key (FK): OwnerOrganizationEmail (OwnerOrganizationEmail references VolunteerOrganization)

Constraints:

- NOT NULL (ID, Content, CreatedDate, OwnerOrganizationEmail)

- UNIQUE (ID)

FinanceCompany

FinanceCompany (

Email: VARCHAR(255)

Name: VARCHAR(255)

Address: VARCHAR(255)

PhoneNumber: CHAR(10)

)

Primary Key (PK): Email

Candidate Key (CK): Email, PhoneNumber

Constraints:

- NOT NULL (Email, Name, Address, PhoneNumber)

- UNIQUE (Email)

AuditCompany

AuditCompany (

Email: VARCHAR(255)

Certification: VARCHAR(255)

AuditMethodology: VARCHAR(255)

)

Primary Key (PK): Email

Candidate Key (CK): Email

Foreign Key (FK): Email (Email references FinanceCompany)

Constraints:

- NOT NULL (Email, Certification, AuditMethodology)

- UNIQUE (Email)

Bank

Bank (

Email: VARCHAR(255)

TotalAssetInApp: DECIMAL(10, 2)

BankType: VARCHAR(255)

)

Primary Key (PK): Email

Candidate Key (CK): Email

Foreign Key (FK): Email (Email references FinanceCompany)

Constraints:

- NOT NULL (Email, TotalAssetInApp, BankType, TrustLevel)

- UNIQUE (Email)

Statement

Statement (

ID: INTEGER

Content: VARCHAR(255)

CreatedDate: DATE

OrgFundId: INTEGER

AuditCompanyEmail: VARCHAR(255)

)

Primary Key (PK): ID

Candidate Key (CK): ID

Foreign Key (FK): ['OrgFundId (OrgFundId references OrganizationFund)', 'AuditCompanyEmail (AuditCompanyEmail references AuditCompany)']

Constraints:

- NOT NULL (ID, Content, CreatedDate, OrgFundId, AuditCompanyEmail)

- UNIQUE (ID)

Fund

Fund (

ID: INTEGER

Purpose: VARCHAR(255)

Balance: DECIMAL(10, 2)

Verification: VARCHAR(255)

BankEmail: VARCHAR(255)

)

Primary Key (PK): ID

Candidate Key (CK): ID

Foreign Key (FK): ['Verification (Verification references Admin)', 'BankEmail (BankEmail references Bank)']

Constraints:

- NOT NULL (ID, Purpose, Balance, Verification, BankEmail)

- UNIQUE (ID)

IndividualFund

IndividualFund (

ID: INTEGER

SSN: INTEGER

)

Primary Key (PK): ID

Candidate Key (CK): ID, SSN

Foreign Key (FK): ID (ID references Fund)

Constraints:

- NOT NULL (ID, SSN)

- UNIQUE (ID)

OrganizationFund

OrganizationFund (

ID: INTEGER

TaxID: INTEGER

)

Primary Key (PK): ID

Candidate Key (CK): ID, TaxID

Foreign Key (FK): ID (ID references Fund)

Constraints:

- NOT NULL (ID, TaxID)

- UNIQUE (ID)

5. Functional Dependencies (FDs):

- Identify the functional dependencies in your relations, including those involving all candidate keys (CK) and primary keys (PK).
- PKs and CKs are considered FDs and should be included in your list of FDs.
- You must include at least one non-PK/CK FD. If you don't have any, you may need to add meaningful attributes to generate valid FDs. Your goal is to engage in a proper normalization exercise.You do not need a non-PK/CK FD for every relation but be reasonable. If the TA feels that non-PK/CK FDs have been omitted, your grade will be adjusted accordingly.

1. **FinanceCompanies Relation**:

- **Attributes**: Email, Name, Address, PhoneNumber
- **Primary Key**： Email
- **Candidate Key:** Email, PhoneNumber
- **FDs**: Email→ Name, Address, PhoneNumber (The finance company's email uniquely determines the name, address and phone number of the finance company)
- **FDs:** Email→Certification, AuditMethodology (AuditCompany Table)
- **FDs:** Email→TotalAssetInApp, BankType, TrustLevel(Bank Table)

2. **Funds Relation**

- **Attributes:** ID, Balance, Purpose
- **Composite Primary Key:** ID
- **FDs:** ID → Balance, Purpose **(**The fund's ID uniquely determines its balance and purpose**)**
- **FDs:** ID → SSN **(**Individual Fund table**)**
- **FDs:** ID → TaxID **(**Organization Fund table**)**

3. **Users Relation**

- **Attributes:** Name, Email, PhoneNumber
- **Primary Key**： Email
- **Candidate Key:** Email

- **FDs**: Email → Name, PhoneNumber (The user's email uniquely determines the name and phone number of the user)
- **FDs**: Email → ActivityHistory (IndividualUser table)
- **FDs**: Email → Name, PhoneNumber (Admin table)

4. **Transactions Relation**:

- **Attributes**: ID, Amount, Content, UserEmail, OrganizationEmail, FundID
- **Primary Key**: ID
- **Candidate Key:** ID
- **FDs**: ID → Amount, Content, UserEmail, OrganizationEmail, FundID (A transaction's ID determines all information of the transaction)

5. **VolunteerOrganization Relation**:

- **Attributes**: Email, Name, Field, Address, Verification
- **Composite Primary Key**： Email
- **Candidate Key:** Email
- **FDs**: Email → Name, Field, Address, Verification (The email uniquely determines the name, address, verification and field of the volunteer organization)

6. **Statements Relation**:

- **Attributes**: ID, Content, CreatedDate, OrgFundId, AuditCompanyEmail
- **Primary Key**: ID
- **Candidate Key:** ID
- **FDs**: ID → Content, CreatedDate, OrgFundId, AuditCompanyEmail (The statement ID determines all information of the statement)

7. Banks Relation (Non PK/CK FD):

- **Attributes**: Bank Type, Total Asset in App, Trust Level
- **FDs**: (Bank Type, Total Asset in App) → Trust Level(The bank type and total asset in app determines its trust level, which will be great for risk assessment)

6. **Normalization**:

- Normalize each table to 3NF or BCNF.
- After normalization, list the tables, their PKs, CKs, and FKs.
- Show the steps taken for the decomposition (similar to the process done in class). Without showing steps, no partial credit will be awarded.

- Format the tables similar to step 4 (`Table1(attr1: domain1, attr2: domain2, ...)`). **All** tables must be listed, not just the ones after normalization.

1. Users Table

Schema: Users(Email, Name, PhoneNumber)

Primary Key (PK): Email

Functional Dependencies (FDs):

Email ⟶ Name, PhoneNumber

Evaluation:

There are no transitive dependencies or partial dependencies. All non-key attributes depend on the primary key.

Result: Already in 3NF.

2. IndividualUser Table

Schema: IndividualUser(Email, ActivityHistory)

PK: Email

CK: Email

FDs:

Email ⟶ ActivityHistory

Evaluation:

No issues; all non-key attributes depend on the primary key.

Result: Already in 3NF.

3. Admin Table

Schema: Admin(Email, VerificationHistory)

PK: Email

CK: Email

FDs:

Email → VerificationHistory

Evaluation:

No issues; all non-key attributes depend on the primary key.

Result: Already in 3NF.

4. VolunteerOrganization Table

Schema: VolunteerOrganization(Email, Name, Field, Address, Verification)

PK: Email

CK: Email

FDs:

Email → Name, Field, Address, Verification

Verification → Email (from Admin)

Evaluation:

All non-key attributes depend on the primary key, but there's a dependency where Verification is a foreign key.

Result: Already in 3NF.

5. Transaction Table

Schema: Transaction(ID, Amount, TransactionDate, Content, UserEmail, OrganizationEmail, FundID)

PK: ID

CK: ID

FDs:

ID → Amount, TransactionDate, Content, UserEmail, OrganizationEmail, FundID

UserEmail → OrganizationEmail (from FK constraints)

Evaluation:

No issues; all non-key attributes depend on the primary key.

Result: Already in 3NF.

6. LawfulDocuments Table

Schema: LawfulDocuments(ID, Content, CreatedDate, OwnerOrganizationEmail)

PK: ID

CK: ID

FDs:

ID ⟶ Content, CreatedDate, OwnerOrganizationEmail

Evaluation:

No issues; all non-key attributes depend on the primary key.

Result: Already in 3NF.

7. FinanceCompany Table

Schema: FinanceCompany(Email, Name, Address, PhoneNumber)

PK: Email

CK: Email, PhoneNumber

FDs:

Email ⟶ Name, Address, PhoneNumber

Evaluation:

No issues; all non-key attributes depend on the primary key.

Result: Already in 3NF.

8. AuditCompany Table

Schema: AuditCompany(Email, Certification, AuditMethodology)

PK: Email

CK: Email

FDs:

Email → Certification, AuditMethodology

Evaluation:

No issues; all non-key attributes depend on the primary key.

Result: Already in 3NF.

9. Bank Table

Schema: Bank(Email, TotalAssetInApp, BankType, TrustLevel)

PK: Email

CK: Email

FDs:

Email → TotalAssetInApp, BankType, TrustLevel

Evaluation:

No issues; all non-key attributes depend on the primary key.

Result: Already in 3NF.

10. Statement Table

Schema: Statement(ID, Content, CreatedDate, OrgFundId, AuditCompanyEmail)

PK: ID

CK: Email

FDs:

ID → Content, CreatedDate, OrgFundId, AuditCompanyEmail

Evaluation:

No issues; all non-key attributes depend on the primary key.

Result: Already in 3NF.

11. Fund Table

Schema: Fund(ID, Purpose, Balance, Verification, BankEmail)

PK: ID

CK: ID

FDs:

ID → Purpose, Balance, Verification, BankEmail

Verification → Email (FK constraint)

Evaluation:

All non-key attributes depend on the primary key, and there's a foreign key constraint on Verification.

Result: Already in 3NF.

12. IndividualFund Table

Schema: IndividualFund(ID, SSN)

PK: ID

CK: ID

FDs:

ID → SSN

Evaluation:

No issues; all non-key attributes depend on the primary key.

Result: Already in 3NF.

13. OrganizationFund Table

Schema: OrganizationFund(ID, TaxID)

PK: ID

CK: ID

FDs:

ID → TaxID

Evaluation:

No issues; all non-key attributes depend on the primary key.

Result: Already in 3NF.

Final Tables and Attributes

Users

- Schema: Users(Email: VARCHAR(255), Name: VARCHAR(50), PhoneNumber: CHAR(10))
- PK: Email
- CK: Email
- FK: None

IndividualUser

- Schema: IndividualUser(Email: VARCHAR(255), ActivityHistory: VARCHAR(255))
- PK: Email
- CK: Email
- FK: Email(Email references Users)

Admin

- Schema: Admin(Email: VARCHAR(255), VerificationHistory: VARCHAR(255))
- PK: Email
- CK: Email
- FK: Email(Email references Users)

VolunteerOrganization

- Schema: VolunteerOrganization(Email: VARCHAR(255), Name: VARCHAR(100), Field: VARCHAR(50), Address: VARCHAR(255), Verification: VARCHAR(255))
- PK: Email
- CK: Email, (Name, Field)
- FK: Verified (Verification references Admin)

Transaction

- Schema: Transaction(ID: INTEGER, Amount: DECIMAL(10, 2), TransactionDate: DATE, Content: VARCHAR(255), UserEmail: VARCHAR(255), OrganizationEmail: VARCHAR(255), FundID: INTEGER)
- PK: ID
- CK: ID
- FK: UserEmail (UserEmail references Users), OrganizationEmail(OrganizationEmail references VolunteerOrganization), FundID(FundID references Fund)

LawfulDocuments

- Schema: LawfulDocuments(ID: NUMBER, Content: VARCHAR(255), CreatedDate: DATE, OwnerOrganizationEmail: VARCHAR(255))
- PK: ID
- CK: ID
- FK: OwnerOrganizationEmail (OwnerOrganizationEmail references VolunteerOrganization)

FinanceCompany

- Schema: FinanceCompany(Email: VARCHAR(255), Name: VARCHAR(255), Address: VARCHAR(255), PhoneNumber: CHAR(10))
- PK: Email
- CK: Email, PhoneNumber
- FK: None

AuditCompany

- Schema: AuditCompany(Email: VARCHAR(255), Certification: VARCHAR(255), AuditMethodology: VARCHAR(255))
- PK: Email
- CK: Email
- FK: Email (Email references FinanceCompany)

Bank

- Schema: Bank(Email: VARCHAR(255), TotalAssetInApp: NUMBER(10, 2), BankType: VARCHAR(255), TrustLevel: NUMBER)
- PK: Email
- CK: Email
- FK: Email(Email references FinanceCompany)

Statement

- Schema: Statement(ID: NUMBER, Content: VARCHAR(255), CreatedDate: DATE, OrgFundId: NUMBER, AuditCompanyEmail: VARCHAR(255))
- PK: ID
- CK: ID
- FK: OrgFundId(OrgFundId references OrganizationFund), AuditCompanyEmail (AuditCompanyEmail references AuditCompany)

Fund

- Schema: Fund(ID: NUMBER, Purpose: VARCHAR(255), Balance: NUMBER(10, 2), Verification: VARCHAR(255), BankEmail: VARCHAR(255))
- PK: ID
- CK: ID
- FK: Verification(Verification references Admin), BankEmail(BankEmail references Bank)

IndividualFund

- Schema: IndividualFund(ID: NUMBER, SSN: NUMBER(9))
- PK: ID
- CK: ID, SSN
- FK: ID(ID references Fund)

OrganizationFund

- Schema: OrganizationFund(ID: NUMBER, TaxID: NUMBER(9))
- PK: ID
- CK: ID, TaxID
- FK: ID (ID references Fund)

6. SQL DDL statements:

```
CREATE TABLE Users (
   Email VARCHAR(255) PRIMARY KEY,
   Name VARCHAR(50) NOT NULL,
   PhoneNumber CHAR(10) NOT NULL,
```

```
   CONSTRAINT email_format CHECK (REGEXP_LIKE(Email,
'^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$'))
);

CREATE TABLE IndividualUser (
   Email VARCHAR(255) PRIMARY KEY,
   ActivityHistory VARCHAR(255),
   CONSTRAINT FK_User FOREIGN KEY (Email) REFERENCES Users(Email)
         ON DELETE CASCADE
);

CREATE TABLE Admin (
   Email VARCHAR(255) PRIMARY KEY,
   VerificationHistory VARCHAR(255),
   CONSTRAINT FK_Admin FOREIGN KEY (Email) REFERENCES Users(Email)
         ON DELETE CASCADE
);
-- Create VolunteerOrganization table
CREATE TABLE VolunteerOrganization (
   Email VARCHAR(255) PRIMARY KEY,
   Name VARCHAR(100) NOT NULL,
   Field VARCHAR(50) NOT NULL,
   Address VARCHAR(255) NOT NULL,
   Verification VARCHAR(255) NOT NULL,
   CONSTRAINT Unique_namefield UNIQUE (Name, Field),
   CONSTRAINT Verified FOREIGN KEY (Verification) REFERENCES Admin(Email)
         ON DELETE CASCADE
);
-- Create Transaction table
CREATE TABLE Transaction (
   ID INTEGER PRIMARY KEY,
   Amount DECIMAL(10, 2) NOT NULL,
   TransactionDate DATE NOT NULL,
   Content VARCHAR(255) NOT NULL,
   UserEmail VARCHAR(255),
   OrganizationEmail VARCHAR(255),
   FundID INTEGER NOT NULL,
   CONSTRAINT Unsigned_Amount CHECK (Amount >= 0),
   CONSTRAINT Tran_Fund FOREIGN KEY (FundID) REFERENCES Fund(ID),
   CONSTRAINT Tran_User FOREIGN KEY (UserEmail) REFERENCES Users(Email) ON
DELETE CASCADE,
   CONSTRAINT Tran_Org FOREIGN KEY (OrganizationEmail) REFERENCES
VolunteerOrganization(Email) ON DELETE CASCADE,
```

```
   CONSTRAINT Tran_Participation CHECK (
      (UserEmail IS NOT NULL AND OrganizationEmail IS NULL) OR
      (UserEmail IS NULL AND OrganizationEmail IS NOT NULL))
);

CREATE TABLE LawfulDocuments (
        ID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
        Content VARCHAR(255) NOT NULL,
        CreatedDate DATE NOT NULL,
        OwnerOrganizationEmail VARCHAR(255) NOT NULL,
        CONSTRAINT FK_Org FOREIGN KEY (OwnerOrganizationEmail) REFERENCES
VolunteerOrganization(Email)
        ON DELETE CASCADE
);



CREATE TABLE FinanceCompany (
        Email VARCHAR(255) PRIMARY KEY,
        Name VARCHAR(255) NOT NULL,
        Address VARCHAR(255) NOT NULL,
        PhoneNumber CHAR(10) UNIQUE NOT NULL
);

CREATE TABLE AuditCompany (
        Email VARCHAR(255) PRIMARY KEY,
        Certification VARCHAR(255) NOT NULL,
        AuditMethodology VARCHAR(255) NOT NULL,
        CONSTRAINT Audit_Fin FOREIGN KEY(Email) REFERENCES
FinanceCompany(Email)
        ON DELETE CASCADE
);

CREATE TABLE Bank (
        Email VARCHAR(255) PRIMARY KEY,
        TotalAssetInApp NUMBER(10, 2) NOT NULL,
        BankType VARCHAR(255) NOT NULL,
        TrustLevel NUMBER NOT NULL,
        CONSTRAINT Bank_Fin FOREIGN KEY(Email) REFERENCES
FinanceCompany(Email),
        CONSTRAINT TotalAsset CHECK (TotalAssetInApp >= 0)
);
```

```
CREATE TABLE Statement (
        ID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
        Content VARCHAR(255) NOT NULL,
        CreatedDate DATE NOT NULL,
        OrgFundId NUMBER NOT NULL,
        AuditCompanyEmail VARCHAR(255) NOT NULL,
        CONSTRAINT Stat_OrgFund FOREIGN KEY(OrgFundId) REFERENCES
OrganizationFund(ID)
        ON DELETE CASCADE,
        CONSTRAINT Stat_Audit FOREIGN KEY(AuditCompanyEmail) REFERENCES
AuditCompany(Email)
        ON DELETE CASCADE
);

CREATE TABLE Fund (
        ID NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
        Purpose VARCHAR(255) NOT NULL,
        Balance NUMBER(10, 2) NOT NULL,
        Verification VARCHAR(255) NOT NULL,
        BankEmail VARCHAR(255) NOT NULL,
        CONSTRAINT Fund_Admin FOREIGN KEY (Verification) REFERENCES Admin(Email)
        ON DELETE CASCADE,
        CONSTRAINT Fund_Bank FOREIGN KEY (BankEmail) REFERENCES Bank(Email)
        ON DELETE CASCADE,
        CONSTRAINT Unsigned_Balance CHECK (Balance >= 0)
);

CREATE TABLE IndividualFund (
        ID NUMBER PRIMARY KEY,
        SSN NUMBER(9) UNIQUE NOT NULL,
        CONSTRAINT Ind_Fund FOREIGN KEY (ID) REFERENCES Fund(ID)
        ON DELETE CASCADE
);

CREATE TABLE OrganizationFund (
        ID NUMBER PRIMARY KEY,
        TaxID NUMBER(9) UNIQUE NOT NULL,
        CONSTRAINT Org_Fund FOREIGN KEY (ID) REFERENCES Fund(ID)
        ON DELETE CASCADE
);
```

IMPORTANT: We use Oracle and it does not support ON UPDATE in schema. As a result, I choose to use a trigger that will be applied:

```
CREATE OR REPLACE TRIGGER update_user_email
BEFORE UPDATE ON Users
FOR EACH ROW
BEGIN
   UPDATE Admin SET Email = :NEW.Email WHERE Email = :OLD.Email;
END;
```

This is a sample trigger for the Users table.

7. INSERT statements:

- Insert at least 5 tuples per table. It's recommended to have more tuples so you can perform meaningful queries later.

```
-- Insert data into Users table
INSERT INTO Users (Email, Name, PhoneNumber)
VALUES
   ('john.doe@gmail.com', 'John Doe', '1234567890'),
   ('jane.smith@yahoo.com', 'Jane Smith', '0987654321'),
   ('michael.brown@outlook.com', 'Michael Brown', '1122334455'),
   ('emily.johnson@live.com', 'Emily Johnson', '2233445566'),
   ('robert.williams@protonmail.com', 'Robert Williams', '3344556677');

-- Insert data into IndividualUser table
INSERT INTO IndividualUser (Email, ActivityHistory)
VALUES
   ('john.doe@gmail.com', 'Participated in community service'),
   ('jane.smith@yahoo.com', 'Volunteered in food bank'),
   ('michael.brown@outlook.com', 'Donated to charity organization'),
   ('emily.johnson@live.com', 'Organized a local charity event'),
   ('robert.williams@protonmail.com', 'Assisted in fundraising event');

-- Insert data into Admin table
INSERT INTO Admin (Email, VerificationHistory)
VALUES
   ('admin.johnson@charity.com', 'Verified 100+ users'),
   ('admin.smith@charity.com', 'Reviewed 50 organization applications'),
   ('admin.doe@charity.com', 'Managed fund allocation'),
   ('admin.williams@charity.com', 'Checked compliance of 30+ transactions'),
```

```
    ('admin.brown@charity.com', 'Supervised event approvals');

-- Insert data into VolunteerOrganization table
INSERT INTO VolunteerOrganization (Email, Name, Field, Address, Verification)
VALUES
    ('save.the.earth@volorg.com', 'Save the Earth', 'Environmental', '123 Greenway Lane',
'admin1@charity.com'),
    ('health.first@volorg.com', 'Health First', 'Healthcare', '456 Health Street',
'admin2@charity.com'),
    ('education.for.all@volorg.com', 'Education for All', 'Education', '789 Learning Ave',
'admin3@charity.com'),
    ('helping.hands@volorg.com', 'Helping Hands', 'Community Support', '321 Help Blvd',
'admin4@charity.com'),
    ('community.care@volorg.com', 'Community Care', 'Social Services', '654 Aid Road',
'admin5@charity.com');

-- Insert data into Transaction table
INSERT INTO Transaction (Amount, TransactionDate, Content, UserEmail, OrganizationEmail,
FundID)
VALUES
    (1000.00, TO_DATE('2024-01-10', 'YYYY-MM-DD'), 'Donation for Environmental Cleanup',
'john.doe@users.com', NULL, 1),
    (500.00, TO_DATE('2024-03-05', 'YYYY-MM-DD'), 'Healthcare Support Contribution', NULL,
'health.first@volorg.com', 2),
    (2000.00, TO_DATE('2024-05-20', 'YYYY-MM-DD'), 'Scholarship Fund Donation',
'jane.smith@users.com', NULL, 3),
    (750.00, TO_DATE('2024-07-15', 'YYYY-MM-DD'), 'Community Support Fund Donation',
NULL, 'community.care@volorg.com', 4),
    (1200.00, TO_DATE('2024-09-05', 'YYYY-MM-DD'), 'Education Fund Donation',
'alice.brown@users.com', NULL, 5);

INSERT INTO LawfulDocuments (Content, CreatedDate, OwnerOrganizationEmail)
VALUES
    ('Charity Legal Compliance Document', TO_DATE('2023-03-10', 'YYYY-MM-DD'),
'save.the.earth@volorg.com'),
    ('Fundraising Authorization Form', TO_DATE('2023-05-15', 'YYYY-MM-DD'),
'health.first@volorg.com'),
    ('Non-Profit Registration Certificate', TO_DATE('2022-12-20', 'YYYY-MM-DD'),
'education.for.all@volorg.com'),
    ('Tax Exemption Declaration', TO_DATE('2024-01-25', 'YYYY-MM-DD'),
'helping.hands@volorg.com'),
    ('Volunteer Organization Governance', TO_DATE('2023-08-17', 'YYYY-MM-DD'),
'community.care@volorg.com');
```

```
INSERT INTO IndividualFund (ID, SSN)
VALUES
   (1, 123456789),
   (2, 987654321),
   (3, 555666777),
   (4, 111222333),
   (5, 444555666);


INSERT INTO OrganizationFund (ID, TaxID)
VALUES
   (1, 100000001),
   (2, 100000002),
   (3, 100000003),
   (4, 100000004),
   (5, 100000005);

INSERT INTO FinanceCompany (Email, Name, Address, PhoneNumber)
VALUES
   ('first.financial@finance.com', 'First Financial Corp', '123 Elm Street, NY', '1234567890'),
   ('golden.trust@finance.com', 'Golden Trust Group', '456 Maple Avenue, LA', '0987654321'),
   ('secure.invest@finance.com', 'Secure Invest LLC', '789 Oak Road, SF', '1122334455'),
   ('capital.growth@finance.com', 'Capital Growth Partners', '321 Cedar Blvd, TX',
'2233445566'),
   ('reliant.financial@finance.com', 'Reliant Financial Services', '654 Pine Street, FL',
'3344556677');

INSERT INTO AuditCompany (Email, Certification, AuditMethodology)
VALUES
   ('first.financial@finance.com', 'ISO 9001', 'Risk-Based Auditing'),
   ('golden.trust@finance.com', 'CPA Certified', 'Compliance Auditing'),
   ('secure.invest@finance.com', 'SOX Certified', 'Operational Auditing'),
   ('capital.growth@finance.com', 'ISO 27001', 'Financial Statement Auditing'),
   ('reliant.financial@finance.com', 'CIA Certified', 'Performance Auditing');

INSERT INTO Bank (Email, TotalAssetInApp, BankType)
VALUES
   ('first.financial@finance.com', 5000000.00, 'Commercial Bank'),
   ('golden.trust@finance.com', 7500000.50, 'Investment Bank'),
   ('secure.invest@finance.com', 3000000.00, 'Savings Bank'),
   ('capital.growth@finance.com', 2500000.75, 'Credit Union'),
   ('reliant.financial@finance.com', 9000000.00, 'Retail Bank');
```

```
INSERT INTO Statement (Content, CreatedDate, OrgFundId, AuditCompanyEmail)
VALUES
   ('Annual Financial Report 2024', TO_DATE('2024-01-15', 'YYYY-MM-DD'), 1,
'first.financial@finance.com'),
   ('Quarterly Compliance Statement', TO_DATE('2024-04-10', 'YYYY-MM-DD'), 2,
'golden.trust@finance.com'),
   ('Tax Audit Results', TO_DATE('2024-07-05', 'YYYY-MM-DD'), 3,
'secure.invest@finance.com'),
   ('End of Year Report 2023', TO_DATE('2023-12-31', 'YYYY-MM-DD'), 4,
'capital.growth@finance.com'),
   ('Mid-Year Performance Audit', TO_DATE('2024-06-20', 'YYYY-MM-DD'), 5,
'reliant.financial@finance.com');
```

```
INSERT INTO Fund (Purpose, Balance, Verification, BankEmail)
VALUES
   ('Scholarship Fund', 50000.00, 'admin1@charity.com', 'first.financial@finance.com'),
   ('Disaster Relief Fund', 150000.75, 'admin2@charity.com', 'golden.trust@finance.com'),
   ('Healthcare Fund', 75000.50, 'admin3@charity.com', 'secure.invest@finance.com'),
   ('Education Fund', 100000.00, 'admin4@charity.com', 'capital.growth@finance.com'),
   ('Community Support Fund', 200000.00, 'admin5@charity.com',
'reliant.financial@finance.com');
```

Notes:

- Be consistent with names used across your ER diagram, schema, and FDs. If names have been intentionally changed, leave a note explaining why.
- You may make any reasonable assumptions regarding missing details but ask your project TA for clarifications if needed.

FAQ:

1. **Testing SQL DDL CREATE and INSERT statements**:
    - Use SQLFiddle for testing. Later, you will learn how to use SQL Plus to run SQL statements from a file.
2. **No FDs beyond primary/candidate keys?**:

    - Identify constraints in your data to create FDs. For example, if you have an Address entity with attributes like House#, Street, City, Province, and Postal Code, create an FD like "Postal Code determines Province".
3. **How many non-PK/CK FDs are needed?**:
    - This depends on your domain. Include as many FDs as the real-world application dictates. If your TA feels you've omitted valid non-PK/CK FDs, your grade may be affected. When in doubt, explain your choices.
4. **Are FDs limited to a single relation?**:
    - FDs capture how values in your data are related. If an FD spans multiple relations, you can still list it, but focus on single-relation FDs for normalization.
5. **Does normalization happen in the real world?**:
    - It depends on factors such as table size, storage space, frequency of reads vs writes, and system hardware. In practice, databases may be heavily normalized or denormalized based on these trade-offs.
6. **Oracle doesn't support ON UPDATE**:
    - Mention in your milestone that you understand it should be "ON UPDATE CASCADE" but Oracle doesn't support it, and explain what alternative you've chosen.
7. **Missing entities/relationships from Milestone 1?**:
    - Yes, there will be penalties if your ER diagram doesn't meet the requirements of Milestone 1, as this will impact subsequent milestones.
8. **Constraints not expressible in SQL DDL?**:
    - Leave a note indicating that you understand the constraints cannot be modeled in SQL DDL and that you will enforce them later.