University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: 7/21/2024

Group Number: 30

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Erwin Nanrey	22597249	u9y1q	caprindersingh@gmail.com
Riu Sugimoto	81972226	x0n9d	agoo140914@gmail.com
Julian Camilo Becerra Leon	22416044	t5c7z	jucble@students.ubc.ca

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

Milestone 2

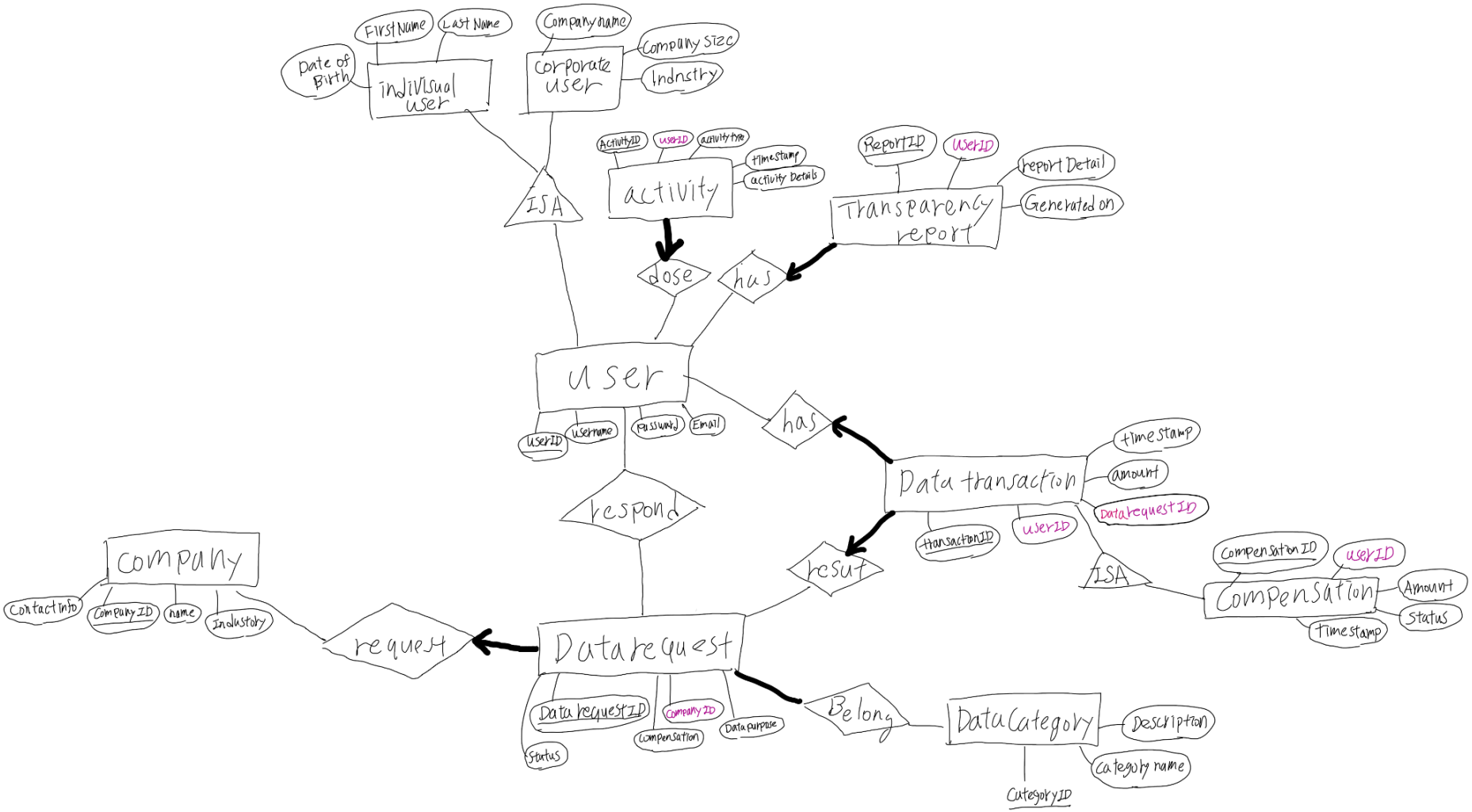
Task 2

Project Description:

A web application that allows users to monetize their personal data, providing transparency and financial compensation when companies utilize the provided user data; the users remain anonymous throughout the entire process. Companies can use the platform to find users with relevant personal data and offer monetary compensation for access to this data. Users can browse through a list of companies looking for specific data, view how much each company is offering per person, and consider the quality of their data as a factor. If users have relevant data to provide, they can click or tap on the listing to begin the process of uploading the required data. Once the user has sent the relevant data, the company has the opportunity to partially review the anonymized data for quality control and, if deemed sufficient, can accept it and pay the anonymized user. The system includes an AI-based component (or a basic version, if AI is not feasible) to provide company recommendations for users based on their listed preferences and, if available, user history. Additionally, the component aids in optimizing data pricing and provides insights for both users and companies. All users receive a transparency report after each successful data transaction, where they can see the information of the company that accepted their data, how much they were paid, what the company plans to do with the data, notes/feedback, etc.

Task 3

Based on the feedback from my TA, the ER diagram was considered overly complicated, which could pose challenges in implementation and management. As a result, we made the decision to simplify the design by removing the Feedback and Payment entities.


First Name
Last Name
Date of Birth
Individual user
Company name
Company Size
Industry
Corporate user
ISA
ActivityID
UserID
activity type
Timestamp
activity Details
activity
dose
ReportID
UserID
report Detail
Generated on
Transparency report
has
user
UserID
username
Password
Email
has
Data transaction
timestamp
amount
DataRequestID
TransactionID
UserID
ISA
CompensationID
UserID
Amount
Status
Timestamp
Compensation
respond
result
company
Contact info
CompanyID
name
Industory
request
Datarequest
DataRequestID
CompanyID
Data purpose
Compensation
Status
Belong
DataCategory
Description
category name
CategoryID

Task 4:

User and IndividualUser / CorporateUser Relationship (ISA):

User(
  UserID: INT,
  UserName: VARCHAR(50),
  Password: INT,
  Email: VARCHAR(50)
)
Constraints:
  Primary Key (PK): UserID
  Not Null: UserID, UserName, Password, Email
  Unique: Email

IndividualUser(
  UserID: INT,
  FirstName: VARCHAR(50),
  LastName: VARCHAR(50),
  DateOfBirth: DATE
)
Constraints:
  Primary Key (PK): UserID
  Foreign Key (FK): UserID references User(UserID) ON DELETE CASCADE
  Not Null: UserID, FirstName, LastName, DateOfBirth
  Candidate Key (CK): (FirstName, LastName, DateOfBirth)

CorporateUser(
  UserID: INT,
  CompanyName: VARCHAR(100),
  Industry: VARCHAR(50),
  CompanySize: INT
)
Constraints:
  Primary Key (PK): UserID
  Foreign Key (FK): UserID references User(UserID) ON DELETE CASCADE
  Not Null: UserID, CompanyName, Industry
  Unique: CompanyName

User and Activity Relationship (one-to-many):

Activity(
  ActivityID: INT,
  UserID: INT,
  ActivityType: VARCHAR(50),
  ActivityDetails: VARCHAR(10000),
  Timestamp: DATE
)
Constraints:
  Primary Key (PK): ActivityID
  Foreign Key (FK): UserID references User(UserID) ON DELETE CASCADE
  Not Null: ActivityID, UserID, ActivityType

User and TransparencyReport Relationship (one-to-many):

TransparencyReport(
  ReportID: INT,
  UserID: INT,
  ReportDetails: VARCHAR(10000),
  GeneratedOn: DATE
)
Constraints:
  Primary Key (PK): ReportID
  Foreign Key (FK): UserID references User(UserID) ON DELETE CASCADE
  Not Null: ReportID, UserID, ReportDetails

User and DataTransaction Relationship (one-to-many):

DataTransaction(
  TransactionID: INT,
  UserID: INT,
  DataRequestID: INT,
  Timestamp: DATE,
  Amount: DECIMAL(10, 2)
)
Constraints:
  Primary Key (PK): TransactionID
  Foreign Key (FK): UserID references User(UserID) ON DELETE CASCADE
  Foreign Key (FK): DataRequestID references DataRequest(DataRequestID)
  Not Null: TransactionID, UserID, DataRequestID

DataRequest and Company Relationship (many-to-one):

Company(
  CompanyID: INT,
  Name: VARCHAR(255),
  Industry: VARCHAR(255),
  ContactInfo: VARCHAR(255)
)
Constraints:
  Primary Key (PK): CompanyID
  Not Null: CompanyID, Name, Industry, ContactInfo
  Unique: Name
  Candidate Key (CK): Name

DataRequest(
  DataRequestID: INT,
  CompanyID: INT,
  Compensation: DECIMAL(10, 2),
  DataPurpose: VARCHAR(255),
  Status: VARCHAR(50)
)
Constraints:
  Primary Key (PK): DataRequestID
  Foreign Key (FK): CompanyID references Company(CompanyID)
  Not Null: DataRequestID, CompanyID, Compensation, DataPurpose, Status

DataRequest and DataCategory Relationship (many-to-many):

DataBelongCategory(
  CategoryID: INT,
  DataRequestID: INT,
  CompanyID: INT
)
Constraints:
  Not Null: CategoryID, DataRequestID, CompanyID
  Primary Key (Composite PK): CategoryID, DataRequestID
  Foreign Key (FK): CategoryID references DataCategory(CategoryID), DataRequestID references DataRequest(DataRequestID), CompanyID references Company(CompanyID)

DataCategory(
  CategoryID: INT,
  CategoryName: VARCHAR(255),
  Description: TEXT
)

Constraints:
  Primary Key (PK): CategoryID
  Unique: CategoryName
  Not Null: Description, CategoryName
  Candidate Key (CK): CategoryName

User and Respond Relationship (many-to-many):

Respond(
  DataRequestID: INT,
  UserID: INT
)
Constraints:
  Primary Key (PK): (DataRequestID, UserID)
  Foreign Key (FK): UserID references User(UserID) ON DELETE CASCADE
  Foreign Key (FK): DataRequestID references DataRequest(DataRequestID) ON DELETE CASCADE
  Not Null: DataRequestID, UserID

DataTransaction and Compensation Relationship (ISA):

Compensation(
  CompensationID: INT,
  UserID: INT,
  TransactionID: INT,
  Amount: DECIMAL(10, 2),
  Timestamp: DATE
)
Constraints:
  Primary Key (PK): CompensationID
  Foreign Key (FK): UserID references User(UserID)
  Foreign Key (FK): TransactionID references DataTransaction(TransactionID)
  Not Null: CompensationID, UserID, TransactionID, Amount, Timestamp

DataTransaction and Compensation Relationship (ISA):

Compensation(
  TransactionID: INT,
  Status: VARCHAR(50),
  Tax: INT,
  NetAmount: INT
)
Constraints:
  Primary Key (PK): TransactionID

Foreign Key (FK): TransactionID references DataTransaction(TransactionID)
Not Null: TransactionID, Status

User and DataTransaction Relationship (one-to-many):

DataTransaction(
TransactionID: INT,
UserID: INT,
DataRequestID: INT,
Amount: INT,
Timestamp: DATE,
Currency: CHAR(3),
ExchangeRate: INT
)
Constraints:
Primary Key (PK): TransactionID
Foreign Key (FK): UserID references User(UserID) ON DELETE CASCADE
Foreign Key (FK): DataRequestID references DataRequest(DataRequestID)
Not Null: TransactionID, UserID, DataRequestID, Amount, Timestamp, Currency, ExchangeRate
Candidate Key (CK): (UserID, DataRequestID)

Task 5:

User:
UserID → Username, Password, Email
Email → Username, Password

IndividualUser :
UserID → FirstName, LastName, DateOfBirth
FirstName, LastName, DateOfBirth → UserID (combination of FirstName, LastName, and DateOfBirth is unique)

CorporateUser :
UserID → CompanyName, Industry, CompanySize
CompanyName → Industry, CompanySize

Activity :
ActivityID → UserID, ActivityType, Timestamp, ActivityDetails
UserID, Timestamp → ActivityType, ActivityDetails

TransparencyReport :
ReportID → UserID, ReportDetails, GeneratedOn
UserID, GeneratedOn → ReportDetails

Compensation:
TransactionID → Status, Tax, NetAmount

DataTransaction:
TransactionID → UserID, DataRequestID, Amount, Timestamp, Currency, ExchangeRate
UserID, DataRequestID → TransactionID, Amount, Timestamp, Currency, ExchangeRate
Currency →ExchangeRate

FDs related to **DataRequest and Company Relationship:**
Company:
CompanyID → Name, Industry, ContactInfo
Name → CompanyID, Industry, ContactInfo (CK)
DataRequest:
DataRequestID → **CompanyID**, Compensation, DataPurpose, Status
CompanyID → Name, Industry, ContactInfo

FD related to **DataRequest and DataCatagory Relationship**:
DataBelongCategory:
CategoryID, **DataRequestID** → **CompanyID**
DataCatagory:
CategoryID → CategoryName**,** Description
CategoryName → CategoryID, Description (CK)

Task 6:

User:

Check if user is in 3NF.
Minimus key of user is UserID .
In order for a relation to be in 3NF, every FD has to have a LHS that is a superkey and/or a RHS that is part of the minimal key. None of the FDs have a RHS that is part of a minimal key. So user is not in 3NF.

decompose into 3NF using the synthesis method.
First, find the minimum cover

1. Put all FDs in standard form.
UserID → Username, *UserID* → Password, *UserID* →Email, Email → Username, Email → Password

2. Minimize the LHS of each FD
Already done

3. Delete redundant FDs
UserID →Email, Email → Username, Email → Password

In synthesis, create a relation for every FD in the minimal cover.
user(*UserID*,Email), user(Email, Username), user(Email,Password)

We add the minimum key user(userID) but itll be the redundant relation so the final answer is
user(*UserID*,Email), user(Email, Username) , user(Email,Password)

IndividualUser :

Check if user is in 3NF.
Minimus key of user is UserID .
In order for a relation to be in 3NF, every FD has to have a LHS that is a superkey and/or a RHS that is part of the minimal key. None of the FDs have a RHS that is part of a minimal key. So user is not in 3NF.

decompose into 3NF using the synthesis method.
First, find the minimum cover

1. Put all FDs in standard form.
UserID → FirstName, *UserID* → LastName, *UserID* →DateOfBirth, FirstName, LastName, DateOfBirth → UserID

2. Minimize the LHS of each FD
UserID → FirstName, *UserID* → LastName, *UserID* →DateOfBirth, FirstName, LastName, DateOfBirth → UserID

3. Delete redundant FDs
UserID → FirstName, *UserID* → LastName, *UserID* →DateOfBirth, FirstName, LastName, DateOfBirth → UserID

In synthesis, create a relation for every FD in the minimal cover.
IndividualUser(*UserID*,FirstName), IndividualUser(*UserID*, LastName),
IndividualUser(*UserID*,DateOfBirth),
IndividualUser(FirstName,LastName,DateOfBirth, UserID)

We add the minimum key IndividualUser(userID) but itll be the redundunt relation so the final anser is

IndividualUser(*UserID*,FirstName), IndividualUser(*UserID*, LastName),
IndividualUser(*UserID*,DateOfBirth),
IndividualUser(FirstName,LastName,DateOfBirth, UserID)

CorporateUser :

Check if user is in 3NF.
Minimus key of user is UserID .
In order for a relation to be in 3NF, every FD has to have a LHS that is a superkey and/or a RHS that is part of the minimal key. None of the FDs have a RHS that is part of a minimal key. So user is not in 3NF.

decompose into 3NF using the synthesis method.
First, find the minimum cover

1. Put all FDs in standard form.
UserID → CompanyName, *UserID* → Industry, *UserID* → CompanySize, CompanyName → Industry, CompanyName → CompanySize

2. Minimize the LHS of each FD
Already done

3. Delete redundant FDs
UserID → CompanyName, CompanyName → Industry, CompanyName → CompanySize

In synthesis, create a relation for every FD in the minimal cover.
CorporateUser (*UserID*, CompanyName), CorporateUser (CompanyName, Industry), CorporateUser(CompanyName, CompanySize)

We add the minimum key user(userID) but itll be the redundant relation so the final answer is
CorporateUser (*UserID*, CompanyName), CorporateUser (CompanyName, Industry), CorporateUser(CompanyName, CompanySize)

Activity :

Check if user is in 3NF.
Minimus key of user is ActivityID .
In order for a relation to be in 3NF, every FD has to have a LHS that is a superkey and/or a RHS that is part of the minimal key. The FD UserID, Timestamp → ActivityType, ActivityDetails does not have a RHS that is part of a minimal key. So user is not in 3NF.

decompose into 3NF using the synthesis method.
First, find the minimum cover

1. Put all FDs in standard form.
ActivityID → UserID, ActivityID → ActivityType, ActivityID→ Timestamp, ActivityID → ActivityDetails,
UserID, Timestamp → ActivityType, UserID, Timestamp → ActivityDetails

2. Minimize the LHS of each FD
ActivityID → UserID, ActivityID → ActivityType, ActivityID→Timestamp, ActivityID → ActivityDetails,
UserID, Timestamp → ActivityType, UserID, Timestamp → ActivityDetails

3. Delete redundant FDs
ActivityID → UserID, ActivityID → Timestamp,
UserID, Timestamp → ActivityType, UserID, Timestamp → ActivityDetails

In synthesis, create a relation for every FD in the minimal cover.
Activity(ActivityID,UserID), Activity(ActivityID ,Timestamp), Activity(UserID, Timestamp ,ActivityType), Activity(UserID, Timestamp ,ActivityDetails)

We add the minimum key user(ActivityID) but it'll be the redundant relation so the final answer is
Activity(ActivityID,UserID), Activity(ActivityID ,Timestamp), Activity(UserID, Timestamp ,ActivityType), Activity(UserID, Timestamp ,ActivityDetails)

TransparencyReport :

ReportID → UserID, ReportDetails, GeneratedOn
UserID, GeneratedOn → ReportDetails

Check if user is in 3NF.
Minimus key of user is ReportID.
In order for a relation to be in 3NF, every FD has to have a LHS that is a superkey and/or a RHS that is part of the minimal key. The FD UserID, GeneratedOn → ReportDetails does not have a RHS that is part of a minimal key. So user is not in 3NF.

decompose into 3NF using the synthesis method.
First, find the minimum cover

1. Put all FDs in standard form.
ReportID → UserID, ReportID → ReportDetails, ReportID →GeneratedOn, UserID, GeneratedOn → ReportDetails

2. Minimize the LHS of each FD
ReportID → UserID, ReportID → ReportDetails, ReportID →GeneratedOn, UserID, GeneratedOn → ReportDetails

3. Delete redundant FDs

ReportID → UserID, ReportID →GeneratedOn, UserID, GeneratedOn → ReportDetails

In synthesis, create a relation for every FD in the minimal cover.
TransparencyReport(ReportID, UserID), TransparencyReport (ReportID, →GeneratedOn), TransparencyReport (UserID, GeneratedOn, ReportDetails)

We add the minimum key user(ReportID) but it'll be the redundant relation so the final answer is
TransparencyReport(ReportID, UserID), TransparencyReport (ReportID, →GeneratedOn), TransparencyReport (UserID, GeneratedOn, ReportDetails)

Compensation:

Check if Compensation is in 3NF.
Key of Compensation is TransactionID.
In order for a relation to be in 3NF, every FD has to have a LHS that is a superkey and/or a RHS that is part of the minimal key.
The only FD has a LHS that is a superkey, or a key in this case. Therefore, Compensation is in 3NF.

DataTransaction:

Check if DataTransaction is in 3NF.
Keys of DataTransaction are (TransactionID) and (UserID, DataRequestID).
In order for a relation to be in 3NF, every FD has to have a LHS that is a superkey and/or a RHS that is part of the minimal key.
In the FD Currency →ExchangeRate, Currency is not a superkey, and ExchangeRate is not part of the minimal key. Therefore, DataTransaction is not in 3NF.

Decompose into 3NF using the synthesis method.
First, find the minimum cover
1. Put all FDs in standard form.
TransactionID → UserID
TransactionID → DataRequestID
TransactionID → Amount
TransactionID → Timestamp
TransactionID → Currency
TransactionID → ExchangeRate
UserID, DataRequestID → TransactionID
UserID, DataRequestID → Amount
UserID, DataRequestID → Timestamp
UserID, DataRequestID → Currency
UserID, DataRequestID → ExchangeRate
Currency → ExchangeRate

2. Minimize the LHS of each FD
TransactionID → UserID
TransactionID → DataRequestID
TransactionID → Amount
TransactionID → Timestamp
TransactionID → Currency
TransactionID → ExchangeRate

TransactionID → DataRequestID
TransactionID → Amount
TransactionID → Timestamp
TransactionID → Currency
TransactionID → ExchangeRate
Currency → ExchangeRate

3. Delete redundant FDs
TransactionID → UserID
TransactionID → DataRequestID
TransactionID → Amount
TransactionID → Timestamp
TransactionID → Currency
TransactionID → ExchangeRate
Currency → ExchangeRate

In synthesis, create a relation for every FD in the minimal cover.
TransactionIDUserID(TransactionID, UserID)
TransactionIDDataRequestID(TransactionID, DataRequestID)
TransactionIDAmount(TransactionID, Amount)
TransactionIDTimestamp(TransactionID, Timestamp)
TransactionIDCurrency(TransactionID, Currency)
TransactionIDExchangeRate(TransactionID, ExchangeRate)
CurrencyExchangeRate(Currency, ExchangeRate)

If there are no relations in the decomposition that contain all of the attributes of a key, add in a relation that contains all the attributes of a key.
Key already present in a relation.

Task 7:

User Table:

```
CREATE TABLE User (
        UserID INTEGER PRIMARY KEY,
        Email VARCHAR(100) NOT NULL,
        UNIQUE (Email)
 );
CREATE TABLE UserEmailUsername (
        Email VARCHAR(100) PRIMARY KEY,
        Username VARCHAR(50) NOT NULL,
        FOREIGN KEY (Email) REFERENCES User(Email)
 );
CREATE TABLE UserEmailPassword (
        Email VARCHAR(100) PRIMARY KEY,
        Password VARCHAR(50) NOT NULL,
```

```
    FOREIGN KEY (Email) REFERENCES User(Email)
);
```

IndividualUser Table:

```
CREATE TABLE IndividualUser (
    UserID INTEGER PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    DateOfBirth DATE NOT NULL
);
CREATE TABLE IndividualUserName (
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    DateOfBirth DATE,
    UserID INTEGER,
    PRIMARY KEY (FirstName, LastName, DateOfBirth),
    FOREIGN KEY (UserID) REFERENCES IndividualUser(UserID)
);
```

CorporateUser Table:

```
CREATE TABLE CorporateUser (
    UserID INTEGER PRIMARY KEY,
    CompanyName VARCHAR(100) NOT NULL
);
CREATE TABLE CompanyNameIndustry (
    CompanyName VARCHAR(100) PRIMARY KEY,
    Industry VARCHAR(50) NOT NULL,
    FOREIGN KEY (CompanyName) REFERENCES CorporateUser(CompanyName)
);

CREATE TABLE CompanyNameSize (
    CompanyName VARCHAR(100) PRIMARY KEY,
    CompanySize VARCHAR(20) NOT NULL,
    FOREIGN KEY (CompanyName) REFERENCES CorporateUser(CompanyName)
);
```

Activity Table:

```
CREATE TABLE Activity (
    ActivityID INTEGER PRIMARY KEY,
    UserID INTEGER NOT NULL,
    FOREIGN KEY (UserID) REFERENCES User(UserID)
);
```

```
CREATE TABLE ActivityTimestamp (
        ActivityID INTEGER PRIMARY KEY,
        Timestamp TIMESTAMP NOT NULL,
        FOREIGN KEY (ActivityID) REFERENCES Activity(ActivityID)
 );

CREATE TABLE UserActivityType (
        UserID INTEGER,
        Timestamp TIMESTAMP,
        ActivityType VARCHAR(50) NOT NULL,
        PRIMARY KEY (UserID, Timestamp),
        FOREIGN KEY (UserID) REFERENCES User(UserID)
 );

CREATE TABLE UserActivityDetails (
        UserID INTEGER,
        Timestamp TIMESTAMP,
        ActivityDetails TEXT,
        PRIMARY KEY (UserID, Timestamp),
        FOREIGN KEY (UserID) REFERENCES User(UserID)
 );
```

TransparencyReport Table:

```
CREATE TABLE TransparencyReport (
        ReportID INTEGER PRIMARY KEY,
        UserID INTEGER NOT NULL,
        FOREIGN KEY (UserID) REFERENCES User(UserID)
 );
CREATE TABLE ReportGeneratedOn (
        ReportID INTEGER PRIMARY KEY,
        GeneratedOn TIMESTAMP NOT NULL,
        FOREIGN KEY (ReportID) REFERENCES TransparencyReport(ReportID)
 );
CREATE TABLE UserGeneratedReportDetails (
        UserID INTEGER,
        GeneratedOn TIMESTAMP,
        ReportDetails TEXT,
        PRIMARY KEY (UserID, GeneratedOn),
        FOREIGN KEY (UserID) REFERENCES User(UserID)
 );
```

Company Table:

```
CREATE TABLE Company (
        CompanyID INT PRIMARY KEY,
        Name VARCHAR(255) NOT NULL UNIQUE,
        Industry VARCHAR(255) NOT NULL,
        ContactInfo VARCHAR(255) NOT NULL
);
```

DataRequest Table:

```
CREATE TABLE DataRequest (
        DataRequestID INT PRIMARY KEY,
        CompanyID INT NOT NULL,
        Compensation DECIMAL(10, 2) NOT NULL,
        DataPurpose VARCHAR(255) NOT NULL,
        Status VARCHAR(50) NOT NULL,
        FOREIGN KEY (CompanyID) REFERENCES Company(CompanyID)
);
```

DataCategory Table:

```
CREATE TABLE DataCategory (
        CategoryID INT PRIMARY KEY,
        CategoryName VARCHAR(255) NOT NULL UNIQUE,
        Description TEXT NOT NULL
);
```

DataBelongCategory Table:

```
CREATE TABLE DataBelongCategory (
        CategoryID INT NOT NULL,
        DataRequestID INT NOT NULL,
        CompanyID INT NOT NULL,
        PRIMARY KEY (CategoryID, DataRequestID),
        FOREIGN KEY (CategoryID) REFERENCES DataCategory(CategoryID),
        FOREIGN KEY (DataRequestID) REFERENCES DataRequest(DataRequestID),
        FOREIGN KEY (CompanyID) REFERENCES Company(CompanyID)
);
```

TransactionIDUserID Table:

```
CREATE TABLE TransactionIDUserID
        (TransactionID INTEGER NOT NULL,
        UserID INTEGER NOT NULL,
        PRIMARY KEY (TransactionID),
        FOREIGN KEY (UserID) REFERENCES User(UserID),
```

```
    UNIQUE (UserID));
```

TransactionIDDataRequestID Table:

```
CREATE TABLE TransactionIDDataRequestID
    (TransactionID INTEGER NOT NULL,
    DataRequestID INTEGER NOT NULL,
    PRIMARY KEY (TransactionID),
    FOREIGN KEY (DataRequestID) REFERENCES DataRequest(DataRequestID),
    UNIQUE (DataRequestID));
```

TransactionIDAmount Table:

```
CREATE TABLE TransactionIDAmount
    (TransactionID INTEGER NOT NULL,
    Amount INTEGER NOT NULL,
    PRIMARY KEY (TransactionID),
    FOREIGN KEY (TransactionID) REFERENCES TransactionIDUserID(TransactionID));
```

TransactionIDTimestamp Table:

```
CREATE TABLE TransactionIDTimestamp
    (TransactionID INTEGER NOT NULL,
    Timestamp DATE NOT NULL,
    PRIMARY KEY (TransactionID),
    FOREIGN KEY (TransactionID) REFERENCES TransactionIDUserID(TransactionID));
```

TransactionIDCurrency Table:

```
CREATE TABLE TransactionIDCurrency
    (TransactionID INTEGER NOT NULL,
    Currency CHAR(3) NOT NULL,
    PRIMARY KEY (TransactionID),
    FOREIGN KEY (TransactionID) REFERENCES TransactionIDUserID(TransactionID));
```

TransactionIDExchangeRate Table:

```
CREATE TABLE TransactionIDExchangeRate
    (TransactionID INTEGER NOT NULL,
    ExchangeRate INTEGER NOT NULL,
    PRIMARY KEY (TransactionID),
    FOREIGN KEY (TransactionID) REFERENCES TransactionIDUserID(TransactionID));
```

CurrencyExchangeRate Table:

```
CREATE TABLE CurrencyExchangeRate
    (Currency CHAR(3) NOT NULL,
    ExchangeRate INTEGER NOT NULL,
```

```
PRIMARY KEY (Currency),
UNIQUE (Currency));
```

Task 8

User Table:

```
INSERT INTO User (UserID, Email) VALUES
 (1, 'alice@example.com'),
 (2, 'bob@example.com'),
 (3, 'carol@example.com'),
 (4, 'dave@example.com'),
 (5, 'eve@example.com');
INSERT INTO UserEmailUsername (Email, Username) VALUES
 ('alice@example.com', 'alice123'),
 ('bob@example.com', 'bobby'),
 ('carol@example.com', 'carol_w'),
 ('dave@example.com', 'davey'),
 ('eve@example.com', 'eve_93');
INSERT INTO UserEmailPassword (Email, Password) VALUES
 ('alice@example.com', 'password1'),
 ('bob@example.com', 'password2'),
 ('carol@example.com', 'password3'),
 ('dave@example.com', 'password4'),
 ('eve@example.com', 'password5');
```

IndividualUser Table:

```
INSERT INTO IndividualUser (UserID, FirstName, LastName, DateOfBirth) VALUES
 (1, 'Alice', 'Smith', '1990-01-01'),
 (2, 'Bob', 'Johnson', '1985-02-02'),
 (3, 'Carol', 'Williams', '1992-03-03'),
 (4, 'Dave', 'Brown', '1988-04-04'),
 (5, 'Eve', 'Jones', '1995-05-05');
INSERT INTO IndividualUserName (FirstName, LastName, DateOfBirth, UserID) VALUES
 ('Alice', 'Smith', '1990-01-01', 1),
 ('Bob', 'Johnson', '1985-02-02', 2),
 ('Carol', 'Williams', '1992-03-03', 3),
 ('Dave', 'Brown', '1988-04-04', 4),
 ('Eve', 'Jones', '1995-05-05', 5);
```

CorporateUser Table:

```
INSERT INTO CorporateUser (UserID, CompanyName) VALUES
 (1, 'TechCorp'),
 (2, 'HealthInc'),
 (3, 'EduSoft'),
 (4, 'FinTech Solutions'),
 (5, 'BioLab');
INSERT INTO CompanyNameIndustry (CompanyName, Industry) VALUES
 ('TechCorp', 'Technology'),
 ('HealthInc', 'Healthcare'),
 ('EduSoft', 'Education'),
 ('FinTech Solutions', 'Finance'),
 ('BioLab', 'Biotechnology');
INSERT INTO CompanyNameSize (CompanyName, CompanySize) VALUES
 ('TechCorp', 'Large'),
 ('HealthInc', 'Medium'),
 ('EduSoft', 'Small'),
 ('FinTech Solutions', 'Large'),
 ('BioLab', 'Medium');
```

Activity Table:

```
INSERT INTO Activity (ActivityID, UserID) VALUES
 (1, 1),
 (2, 2),
 (3, 3),
 (4, 4),
 (5, 5);
INSERT INTO ActivityTimestamp (ActivityID, Timestamp) VALUES
 (1, '2024-07-01 10:00:00'),
 (2, '2024-07-01 11:00:00'),
 (3, '2024-07-01 12:00:00'),
 (4, '2024-07-01 13:00:00'),
 (5, '2024-07-01 14:00:00');
INSERT INTO UserActivityType (UserID, Timestamp, ActivityType) VALUES
 (1, '2024-07-01 10:00:00', 'Login'),
 (2, '2024-07-01 11:00:00', 'Data Upload'),
 (3, '2024-07-01 12:00:00', 'Data Download'),
 (4, '2024-07-01 13:00:00', 'Logout'),
 (5, '2024-07-01 14:00:00', 'Profile Update');
INSERT INTO UserActivityDetails (UserID, Timestamp, ActivityDetails) VALUES
 (1, '2024-07-01 10:00:00', 'User logged in from IP 192.168.1.1'),
 (2, '2024-07-01 11:00:00', 'Uploaded dataset file1.csv'),
 (3, '2024-07-01 12:00:00', 'Downloaded report file2.pdf'),
```

(4, '2024-07-01 13:00:00', 'User logged out successfully'),
(5, '2024-07-01 14:00:00', 'Updated profile picture');

TransparencyReport Table:

INSERT INTO TransparencyReport (ReportID, UserID) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5);
INSERT INTO ReportGeneratedOn (ReportID, GeneratedOn) VALUES
(1, '2024-07-01 15:00:00'),
(2, '2024-07-01 16:00:00'),
(3, '2024-07-01 17:00:00'),
(4, '2024-07-01 18:00:00'),
(5, '2024-07-01 19:00:00');
INSERT INTO UserGeneratedReportDetails (UserID, GeneratedOn, ReportDetails) VALUES
(1, '2024-07-01 15:00:00', 'Report on data usage for July 2024'),
(2, '2024-07-01 16:00:00', 'Report on data transactions for July 2024'),
(3, '2024-07-01 17:00:00', 'Report on user activity for July 2024'),
(4, '2024-07-01 18:00:00', 'Report on financial transactions for July 2024'),
(5, '2024-07-01 19:00:00', 'Report on profile updates for July 2024');

Company Table:

INSERT INTO Company (CompanyID, Name, Industry, ContactInfo) VALUES
(1, 'TechCorp', 'Technology', 'contact@techcorp.com'),
(2, 'HealthPlus', 'Healthcare', 'info@healthplus.com'),
(3, 'EduWorld', 'Education', 'support@eduworld.com'),
(4, 'FinancePros', 'Finance', 'contact@financepros.com'),
(5, 'EcoLiving', 'Environment', 'info@ecoliving.com');

DataRequest Table:

INSERT INTO DataRequest (DataRequestID, CompanyID, Compensation, DataPurpose, Status) VALUES
(1, 1, 1000.00, 'Market Research', 'Pending'),
(2, 2, 1500.00, 'Clinical Study', 'Approved'),
(3, 3, 500.00, 'Educational Analysis', 'Pending'),
(4, 4, 2000.00, 'Financial Forecast', 'Rejected'),
(5, 5, 1200.00, 'Environmental Impact Study', 'Approved');

DataCategory Table:

INSERT INTO DataCategory (CategoryID, CategoryName, Description) VALUES

(1, 'Demographics', 'Data related to population demographics'),
(2, 'Health', 'Health-related data'),
(3, 'Education', 'Educational data'),
(4, 'Finance', 'Financial data'),
(5, 'Environment', 'Environmental data');

DataBelongCategory Table:

INSERT INTO DataBelongCategory (CategoryID, DataRequestID, CompanyID) VALUES
(1, 1, 1),
(2, 2, 2),
(3, 3, 3),
(4, 4, 4),
(5, 5, 5);

TransactionIDUserID Table:

INSERT INTO TransactionIDUserID (TransactionID, UserID) VALUES
(1, 101),
(2, 102),
(3, 103),
(4, 104),
(5, 105);

TransactionIDDataRequestID Table:

INSERT INTO TransactionIDDataRequestID (TransactionID, DataRequestID) VALUES
(1, 201),
(2, 202),
(3, 203),
(4, 204),
(5, 205);

TransactionIDAmount Table:

INSERT INTO TransactionIDAmount (TransactionID, Amount) VALUES
(1, 500),
(2, 1500),
(3, 2500),
(4, 3500),
(5, 4500);

TransactionIDTimestamp Table:

INSERT INTO TransactionIDTimestamp (TransactionID, Timestamp) VALUES
(1, '2024-01-01'),
(2, '2024-01-02'),

(3, '2024-01-03'),
(4, '2024-01-04'),
(5, '2024-01-05');

TransactionIDCurrency Table:

INSERT INTO TransactionIDCurrency (TransactionID, Currency) VALUES
(1, 'USD'),
(2, 'EUR'),
(3, 'GBP'),
(4, 'JPY'),
(5, 'CAD');

TransactionIDExchangeRate Table:

INSERT INTO TransactionIDExchangeRate (TransactionID, ExchangeRate) VALUES
(1, 1),
(2, 0.9),
(3, 0.8),
(4, 110),
(5, 1.3);

CurrencyExchangeRate Table:

INSERT INTO CurrencyExchangeRate (Currency, ExchangeRate) VALUES
('USD', 1),
('EUR', 0.9),
('GBP', 0.8),
('JPY', 110),
('CAD', 1.3);