University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: Oct 20, 2023

Group Number: 30

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Andrew Fenton	43711555	w7a1m	andrewfenton898@gmail.com
Ben Vinnick	20038972	s7f3b	benvinnick@gmail.com
Cameron McInnes	61765640	k2h3b	camtmcinnes@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.)

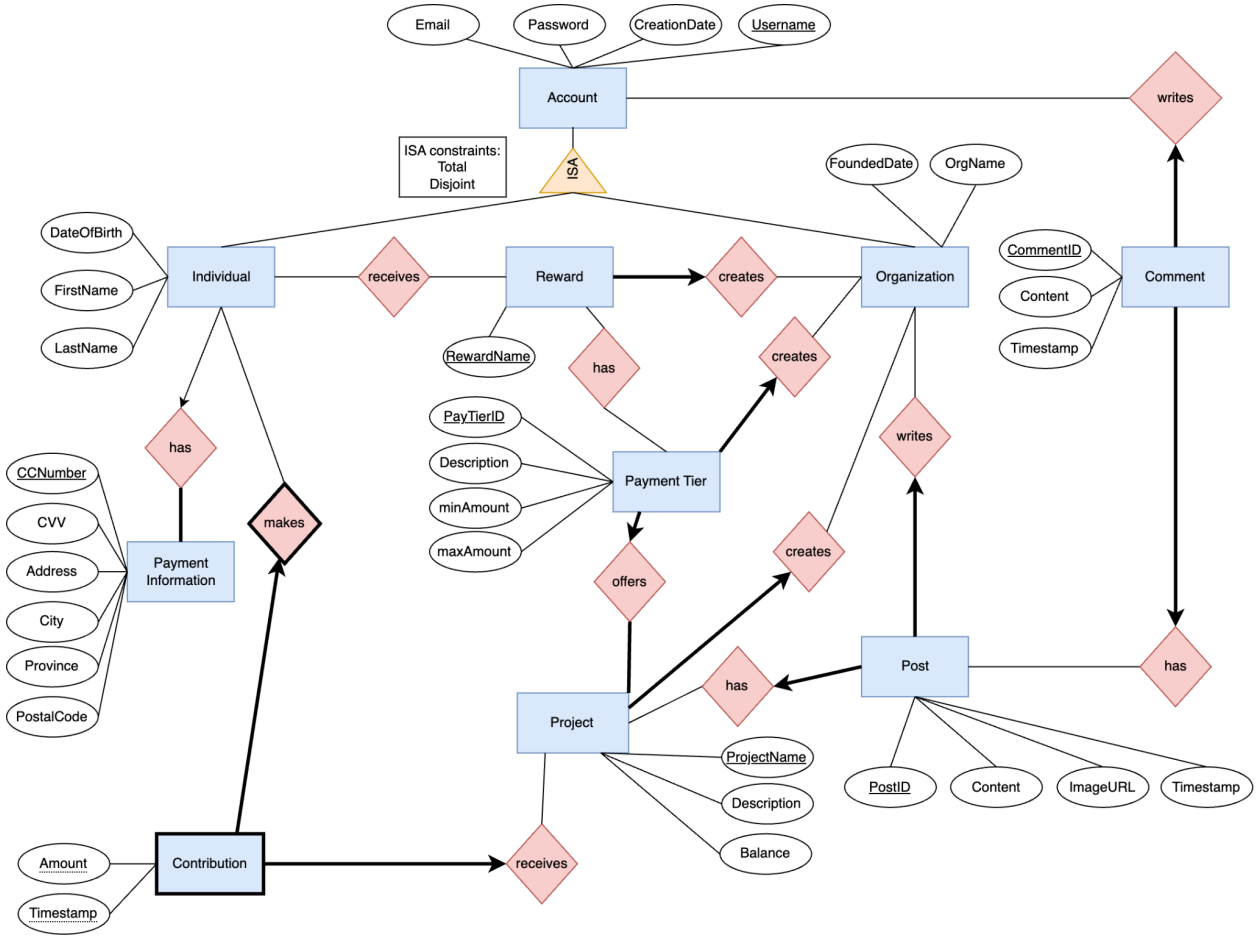
In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Project Summary

Our project simulates a crowdfunding platform. In this platform, organizations can create projects that individuals can support through financial contributions–they may also receive a reward for their donation. Each organization can make dedicated posts for any particular project they own and any type of account can comment on these posts.

ER Diagram

[Link to full size image]



Changes to ER Diagram:

- We changed the attribute email to be UNIQUE and made username the primary key for simplicity. We had too many tables that relied on unique IDs for their primary keys. It's simpler to just use a name (if possible) so we changed Reward to use RewardName and Project to use ProjectName for their primary keys instead of using a unique ID. (suggested by TA).
- We removed the timestamp attribute from the “writes” relationship between Organization and Post to just be an attribute in the Post entity. We also removed the timestamp attribute from the “has” relationship between Comment and Post to just be an attribute in the Comment entity. (suggested by TA)

- We changed the relationship “has” between Individual and PaymentInformation into a many to one relationship so that the same payment information can be used for more than one individual. This for example would allow families to share the same payment information (suggested by the TA). We also changed the total participation so that an individual does not require any payment information. If they choose
- We changed the weak entity Contribution to only have one owner, Individual, since it does not follow the class' ER diagram design rules (suggested by TA)

Parts 4, 5, 6.

Relational Model	Functional Dependencies	Normalization
Account (Username: VARCHAR(50), Password: VARCHAR(50), CreationDate: DATE Email: VARCHAR(50)) ➔ User must have a password to login (NOT NULL) ➔ Email is UNIQUE and NOT NULL (Candidate Key)	Username -> Email, Password, CreationDate Email -> Username	Is in BCNF. The closures of all functional dependencies encapsulate all the attributes in the relation.

PaymentInformation(CCNumber: CHAR(16), CVV: CHAR(3), Address: VARCHAR(50), City: VARCHAR(50), Province: VARCHAR(50), PostalCode: CHAR(6)) ➔ CVV, Address, City, Province, PostalCode NOT NULL (need to know this to make a donation) ➔ Here we need to add an assertion to make sure each PaymentInformation is referenced by at least one Individual (since it has total participation constraint)	CCNumber -> CVV, Address, City, Province, PostalCode PostalCode -> City, Province	Is not in BCNF **Decomposed Relations:** PaymentInformation(CCNumber: CHAR(16), CVV: CHAR(3), Address: VARCHAR(50), **PostalCode**: CHAR(6)) ➔ CVV, Address, PostalCode NOT NULL PostalCode_Location(PostalCode: CHAR(6), City: VARCHAR(50), Province: VARCHAR(50)) ➔ City, Province NOT NULL **See justification below.**
Individual (**Username:** VARCHAR(50), **PaymentInfo:** CHAR(16), DateOfBirth: DATE, FirstName: VARCHAR(50), LastName: VARCHAR(50))	Username -> DateOfBirth, FirstName, LastName, PaymentInfo	Is in BCNF

Organization (**Username:** VARCHAR(50), FoundedDate: DATE, OrgName: VARCHAR(50)) ➔ OrgName is UNIQUE and NOT NULL (Candidate Key)	Username -> OrgName, FoundedDate OrgName -> Username	Is in BCNF
Organization_creates_Project (ProjectName: VARCHAR(50), **OUsername**: VARCHAR(50), Description: VARCHAR(200), Balance: INT) ➔ OUsername is NOT NULL ➔ Balance >= 0 and NOT NULL	ProjectName -> OUsername, Description, Name, Balance	Is in BCNF
Organization_creates_Post (PostID:INT, **OUsername:** VARCHAR(50), **ProjectName**: VARCHAR(50), Content: VARCHAR(600), ImageURL: VARCHAR(200), Timestamp: TIMESTAMP) ➔ OUsername NOT NULL ➔ ProjectName NOT NULL ➔ Each post needs content so content is NOT NULL	PostID -> OUsername, ProjectName, Content, ImageURL, Timestamp	Is in BCNF

Account_writes_Comment_on_Post (CommentID: INT, **Username:** VARCHAR(50), **PostID: INT**, Timestamp: TIMESTAMP, Content: VARCHAR(200)) ➔ Username NOT NULL ➔ PostID NOT NULL ➔ Each comment needs content so content is NOT NULL	CommentID -> Username, PostID, Timestamp, Content	Is in BCNF
Individual_makes_Contribution (**IUsername:** VARCHAR(50), **ProjectName:** VARCHAR(50), Amount: INT, Timestamp: TIMESTAMP) ➔ ProjectName is NOT NULL ➔ Amount is > 0 and NOT NULL	IUsername, Amount, Timestamp -> ProjectName	Is in BCNF
Reward (RewardName: VARCHAR(50), **OUsername:** VARCHAR(50)) ➔ OUsername is NOT NULL	RewardName -> OUsername	Is in BCNF
Individual_receives_Reward(**RewardName:** INT, **IUsername**: VARCHAR(50),)	Implicit FDs only.	Is in BCNF

PaymentTier(PayTierID: INT, **ProjectName:** VARCHAR(50), **OUsername:** VARCHAR(50), Description: VARCHAR(100), minAmount: INT, maxAmount: INT) ➔ ProjectName, OUsername are NOT NULL ➔ minAmount and maxAmount > 0 & NOT NULL ➔ minAmount <= maxAmount	PayTierId -> ProjectName, OUsername, Description, minAmount, maxAmount	Is in BCNF
PaymentTier_has_Reward (**PayTierID**: INT, **RewardName**: VARCHAR(50))	Implicit FDs only.	Is in BCNF

Justification for Decomposition of PaymentInformation

Every table except Individual_has_PaymentInformation is in BCNF (no FDs in violation).

Individual_has_PaymentInformation is not in BCNF because PostalCode -> City, Province violates the rule for BCNF (Given FD PostalCode -> City, Province: PostalCode is not a superkey!)

Normalize Individual_has_PaymentInformation into BCNF:

Payment_Information:

CCNumber -> CVV, Address, City, Province, PostalCode

PostalCode -> City, Province

CCNumber+ = {CCNumber, CVV, Address, City, Province, PostalCode}
PostalCode+ = {PostalCode, City, Province}

Is it in BCNF? No, because the FD: PostalCode -> City, Province, does not hold for the relation since PostalCode is not a super key.

Decompose on PostalCode -> City, Province

R1(<u>CCNumber</u>, CVV, Address, **PostalCode**)
R2 (<u>PostalCode</u>, City, Province)

Parts 6 & 7.

Normalized Relations	**SQL DDL Statements**
Account (Username: VARCHAR(50), Password: VARCHAR(50), CreationDate: DATE Email: VARCHAR(50)) ➔ User must have a password to login (NOT NULL) ➔ Email is UNIQUE and NOT NULL (Candidate Key)	CREATE TABLE Account (Username VARCHAR(50) PRIMARY KEY, Password VARCHAR(50) NOT NULL, CreationDate DATE, Email VARCHAR(50) UNIQUE NOT NULL); – User must have a password to login (NOT NULL)
PostalCode_Location(PostalCode: CHAR(6), City: VARCHAR(50), Province: VARCHAR(50)) ➔ City and province must be NOT NULL since they are used by PaymentInformation for payments	CREATE TABLE PostalCode_Location (PostalCode CHAR(6) PRIMARY KEY, City VARCHAR(50) NOT NULL, Province VARCHAR(50) NOT NULL); – Here we need to add an assertion to make sure a PostalCode_Location is referenced by at least one Payment Information – City and province must be NOT NULL since they are used by PaymentInformation for payments

PaymentInformation(CCNumber: CHAR(16), CVV: CHAR(3), Address: VARCHAR(50), **PostalCode**: CHAR(6)) ➔ Here we need to add an assertion to make sure each PaymentInformation is referenced by at least one Individual (since it has total participation constraint) ➔ – CVV, Address are required for any payments (NOT NULL)	CREATE TABLE PaymentInformation(CCNumber CHAR(16) PRIMARY KEY, CVV CHAR(3) NOT NULL, Address VARCHAR(50) NOT NULL, PostalCode CHAR(6) NOT NULL, FOREIGN KEY(PostalCode) REFERENCES PostalCode_Location ON DELETE NO ACTION); – Here we need to add an assertion to make sure a PaymentInformation is referenced by at least one Individual (since it has total participation constraint) – Oracle doesn't support ON UPDATE so we will need to use something like a trigger to update PaymentInformation when PostalCode_Location is updated – ON DELETE: prevent postal code from being deleted if any payment information references it – CVV, Address are required for any payments (NOT NULL)
Individual (**Username:** VARCHAR(50), **PaymentInfo:** CHAR(16), DateOfBirth: DATE, FirstName: VARCHAR(50), LastName: VARCHAR(50))	CREATE TABLE Individual(Username VARCHAR(50) PRIMARY KEY, PaymentInfo CHAR(16), DateOfBirth DATE, FirstName VARCHAR(50), LastName VARCHAR(50), FOREIGN KEY(Username) REFERENCES Account ON DELETE CASCADE, FOREIGN KEY(PaymentInfo) REFERENCES PaymentInformation ON DELETE SET NULL); – Oracle doesn't support ON UPDATE so we will need to use something like a trigger to update Individual when Account is updated, or when PaymentInformation is Updated – ON DELETE: If an account is deleted, delete the individual associated with that account

Organization (**<u>Username</u>:** VARCHAR(50), FoundedDate: DATE, OrgName: VARCHAR(50)) ➔ OrgName is UNIQUE and NOT NULL (Candidate Key)	CREATE TABLE Organization(Username VARCHAR(50) PRIMARY KEY, FoundedDate DATE, OrgName VARCHAR(50) UNIQUE NOT NULL, FOREIGN KEY(Username) REFERENCES Account ON DELETE CASCADE); – Oracle doesn't support ON UPDATE so we will need to use something like a trigger to update Organization when Account is updated – ON DELETE: If an account is deleted, delete the organization associated with that account
Organization_creates_Project (<u>ProjectName</u>: VARCHAR(50), **OUsername**: VARCHAR(50), Description: VARCHAR(200), Name: VARCHAR(100), Balance: INT) ➔ OUsername is NOT NULL ➔ Balance >= 0 and NOT NULL	CREATE TABLE Organization_creates_Project (ProjectName VARCHAR(50) PRIMARY KEY, OUsername VARCHAR(50) NOT NULL, Description VARCHAR(200), Name VARCHAR(100), Balance INT NOT NULL, FOREIGN KEY(OUsername) REFERENCES Organization ON DELETE NO ACTION); – Here we need to add an assertion to make sure a Project is referenced by at least one Payment Tier – Oracle doesn't support ON UPDATE so we will need to use something like a trigger to update Organization_creates_Project when Organization is updated – Oracle doesn't support unsigned ints so we will need to add a constraint for Balance >= 0. Must also have a value (NOT NULL). – ON DELETE: prevent an organization from being deleted if it has any projects referencing it. Many people may have made contributions that we do not want to disappear.

Organization_creates_Post (PostID: INT, **OUsername:** VARCHAR(50), **ProjectName**: VARCHAR(50), Content: VARCHAR(600), ImageURL: VARCHAR(200), Timestamp: TIMESTAMP) ➔ OUsername NOT NULL ➔ ProjectName NOT NULL ➔ A post must have some associated text. Makes no sense to post nothing (content is NOT NULL)	CREATE TABLE Organization_creates_Post (PostID INT PRIMARY KEY, OUsername VARCHAR(50) NOT NULL, ProjectName VARCHAR(50) NOT NULL, Content VARCHAR(600) NOT NULL, ImageURL VARCHAR(200), Timestamp TIMESTAMP, FOREIGN KEY(OUsername) REFERENCES Organization, FOREIGN KEY(ProjectName) REFERENCES Organization_creates_Project ON DELETE CASCADE); – Oracle doesn't support ON UPDATE so we will need to use something like a trigger to update Organization_creates_Post when Organization is updated, or when Organization_creates_Project is updated – ON DELETE: if the associated project is deleted, delete the post. We don’t need to worry about on delete for organization because an organization can’t be deleted if it has a project, and without a project we wouldn’t have this post. – A post must have some associated text. Makes no sense to post nothing (content is NOT NULL)

Account_writes_Comment_on_Post (CommentID: INT, **Username:** VARCHAR(50), **PostID: INT**, Timestamp: TIMESTAMP, Content: VARCHAR(200)) ➔ Username NOT NULL ➔ PostID NOT NULL ➔ A comment must have some associated text (content is NOT NULL)	CREATE TABLE Account_writes_Comment_on_Post (CommentID INT PRIMARY KEY, Username VARCHAR(50) NOT NULL, PostID INT NOT NULL, Timestamp TIMESTAMP, Content VARCHAR(200) NOT NULL, FOREIGN KEY(Username) REFERENCES Account ON DELETE CASCADE, FOREIGN KEY(PostID) REFERENCES Organization_creates_Post ON DELETE CASCADE); – Oracle doesn't support ON UPDATE so we will need to use something like a trigger to update Account_writes_Comment_on_Post when Account is updated, or when Organization_creates_Post is updated – ON DELETE: If an account or post is deleted, all associated comments are deleted. – A comment must have some associated text (content is NOT NULL)

Individual_makes_Contribution (**IUsername:** VARCHAR(50), **ProjectName:** VARCHAR(50), Amount: INT, Timestamp: TIMESTAMP) ➔ ProjectName is NOT NULL ➔ Amount is > 0 ➔ A donation must have an amount (NOT NULL)	CREATE TABLE Individual_makes_Contribution (IUsername VARCHAR(50), ProjectName VARCHAR(50) NOT NULL, Amount INT NOT NULL, Timestamp TIMESTAMP, PRIMARY KEY(IUsername, Amount, Timestamp), FOREIGN KEY(IUsername) REFERENCES Individual ON DELETE NO ACTION , FOREIGN KEY(ProjectName) REFERENCES Organization_creates_Project ON DELETE CASCADE); – Oracle doesn't support ON UPDATE so we will need to use something like a trigger to update Individual_makes_contribution when Individual is updated, or when Organization_creates_Project is updated – Oracle doesn't support unsigned ints so we will need to add a constraint for amount > 0. A donation must have an amount (NOT NULL) – ON DELETE: prevent deletion of individuals when they have made any contributions. Deleting a project will delete the associated contributions it received from record.
Reward (RewardName: VARCHAR(50), **OUsername:** VARCHAR(50)) ➔ OUsername is NOT NULL	CREATE TABLE Reward (RewardName VARCHAR(50), OUsername VARCHAR(50), PRIMARY KEY(RewardName), FOREIGN KEY(OUsername) REFERENCES Organization ON DELETE CASCADE); – Oracle doesn't support ON UPDATE so we will need to use something like a trigger to update Reward when Organization is updated – ON DELETE: if an organization is deleted, any associated rewards will be deleted (assuming that no individual has received the reward).

Individual_receives_Reward(**RewardName:** VARCHAR(50), **IUsername**: VARCHAR(50),)	CREATE TABLE Individual_receives_Reward (RewardName VARCHAR(50), IUsername VARCHAR(50), PRIMARY KEY(RewardName, IUsername), FOREIGN KEY(RewardName) REFERENCES Reward ON DELETE NO ACTION, FOREIGN KEY(IUsername) REFERENCES Individual ON DELETE CASCADE); - ON DELETE: prevent issued rewards from being deleted if that organization tries to delete the reward. If a user deletes their account, their associated issued rewards will be deleted.
PaymentTier(PayTierID: INT, **ProjectName:** VARCHAR(50), **OUsername:** VARCHAR(50), Description: VARCHAR(100), minAmount: INT, maxAmount: INT) ➔ ProjectName, OUsername are NOT NULL ➔ minAmount and maxAmount > 0 ➔ minAmount <= maxAmount ➔ minAmount and maxAmount are NOT NULL	CREATE TABLE PaymentTier(PayTierID INT PRIMARY KEY, ProjectName VARCHAR(50) NOT NULL, OUsername VARCHAR(50) NOT NULL, Description VARCHAR(100), minAmount INT NOT NULL, maxAmount INT NOT NULL, FOREIGN KEY(ProjectName) REFERENCES Organization_creates_Project ON DELETE CASCADE, FOREIGN KEY(OUsername) REFERENCES Organization ON DELETE CASCADE); – Oracle doesn't support ON UPDATE so we will need to use something like a trigger to update PaymentTier when Organization_creates_Project is updated, or when Organization is updated – Oracle doesn't support unsigned ints so we will need to add a constraint for minAmount > 0, maxAmount >= 0 & minAmount <= maxAmount. Both these fields also must be NOT NULL so that Individuals know how much they can donate. – ON DELETE: If a project is deleted, delete the associated payment tiers. Upon organization deletion, remove associated payment tiers.

PaymentTier_has_Reward (**PayTierID**: INT, **RewardName**: VARCHAR(50))	CREATE TABLE PaymentTier_has_Reward (PayTierID INT, RewardName VARCHAR(50), PRIMARY KEY(PayTierID, RewardName), FOREIGN KEY(PayTierID) REFERENCES PaymentTier ON DELETE CASCADE, FOREIGN KEY(RewardName) REFERENCES Reward ON DELETE CASCADE); – Oracle doesn't support ON UPDATE so we will need to use something like a trigger to update PaymentTier_has_Reward when PaymentTier is updated, or when Reward is updated - ON DELETE: When a PaymentTier or a Reward is deleted we want the rows referencing the PaymentTier or Reward to be deleted as well.

Insert statements for Account

```
INSERT INTO Account (Username, Password, CreationDate, Email)
	VALUES ('cmcdavid1', 'Cpass1', TO_DATE('2023-03-15', 'YYYY-MM-DD'),
'cmcdavid@gmail.com');

INSERT INTO Account (Username, Password, CreationDate, Email)
	VALUES ('ntesla12', 'Npass3', TO_DATE('2023-04-22', 'YYYY-MM-DD'),
'ntesla@gmail.com');

INSERT INTO Account (Username, Password, CreationDate, Email)
	VALUES ('aeinstein30', 'Apass7', TO_DATE('2023-06-10', 'YYYY-MM-DD'),
'aeinstein@gmail.com');

INSERT INTO Account (Username, Password, CreationDate, Email)
	VALUES ('pshelby1', 'Ppass2', TO_DATE('2023-07-29', 'YYYY-MM-DD'),
'pshelby@gmail.com');

INSERT INTO Account (Username, Password, CreationDate, Email)
	VALUES ('tshebs', 'Tpass9', TO_DATE('2023-08-18', 'YYYY-MM-DD'),
'tshelby@gmail.com');

INSERT INTO Account (Username, Password, CreationDate, Email)
	VALUES ('ashebs12', 'Artpass9', TO_DATE('2023-08-18', 'YYYY-MM-DD'),
'artshelby@gmail.com');

INSERT INTO Account (Username, Password, CreationDate, Email)
	VALUES ('nomoney10', 'nmonpass10', TO_DATE('2023-08-20', 'YYYY-MM-DD'),
'igotnomoney@gmail.com');

INSERT INTO Account (Username, Password, CreationDate, Email)
	VALUES ('wildlife', 'wlifepass10', TO_DATE('2023-01-18', 'YYYY-MM-DD'),
'wlsaver@gmail.com');

INSERT INTO Account (Username, Password, CreationDate, Email)
	VALUES ('globewarm', 'gwarmpass1', TO_DATE('2023-04-12', 'YYYY-MM-DD'),
'gwarm@gmail.com');

INSERT INTO Account (Username, Password, CreationDate, Email)
	VALUES ('ubcneedsmoney', 'ubcpass1', TO_DATE('2023-10-05', 'YYYY-MM-DD'),
'ubc@gmail.com');
```

```
INSERT INTO Account (Username, Password, CreationDate, Email)
	VALUES ('sfuneedsmoney', 'sfupass1', TO_DATE('2023-10-12', 'YYYY-MM-DD'),
'sfu@gmail.com');

INSERT INTO Account (Username, Password, CreationDate, Email)
	VALUES ('fundbenslife', 'fbenpass', TO_DATE('2023-07-12', 'YYYY-MM-DD'),
'fundben@gmail.com');
```

Insert statements for PostalCode_Location

```
INSERT INTO PostalCode_Location (PostalCode, City, Province)
	VALUES ('M5H1W7', 'Toronto', 'ON');

INSERT INTO PostalCode_Location (PostalCode, City, Province)
	VALUES ('H2X1L4', 'Montreal', 'QC');

INSERT INTO PostalCode_Location (PostalCode, City, Province)
	VALUES ('V6Z1K7', 'Vancouver', 'BC');

INSERT INTO PostalCode_Location (PostalCode, City, Province)
	VALUES ('E1C4P9', 'Moncton', 'NB');

INSERT INTO PostalCode_Location (PostalCode, City, Province)
	VALUES ('L5N2X2', 'Mississauga', 'ON');
```

Insert statements for PaymentInformation

```
INSERT INTO PaymentInformation (CCNumber, CVV, Address, PostalCode)
	VALUES ('1111222233334444', '123', '123 Main St', 'M5H1W7');

INSERT INTO PaymentInformation (CCNumber, CVV, Address, PostalCode)
	VALUES ('2222333344445555', '456', '456 Elm St', 'H2X1L4');

INSERT INTO PaymentInformation (CCNumber, CVV, Address, PostalCode)
	VALUES ('3333444455556666', '789', '789 Oak St', 'V6Z1K7');

INSERT INTO PaymentInformation (CCNumber, CVV, Address, PostalCode)
	VALUES ('4444555566667777', '234', '234 Pine St', 'E1C4P9');

INSERT INTO PaymentInformation (CCNumber, CVV, Address, PostalCode)
	VALUES ('5555666777788888', '567', '567 Cedar St', 'L5N2X2');

INSERT INTO PaymentInformation (CCNumber, CVV, Address, PostalCode)
	VALUES ('1234123412341234', '111', '111 Cedar St', 'L5N2X2');
```

Insert statements for Individual

```
INSERT INTO Individual (Username, PaymentInfo, DateOfBirth, FirstName, LastName)
	VALUES ('cmcdavid1', '3333444455556666', TO_DATE('1990-05-15', 'YYYY-MM-DD'),
'Connor', 'McDavid');

INSERT INTO Individual (Username, PaymentInfo, DateOfBirth, FirstName, LastName)
	VALUES ('ntesla12', '4444555566667777', TO_DATE('1950-01-12', 'YYYY-MM-DD'),
'Nikola', 'Tesla');

INSERT INTO Individual (Username, PaymentInfo, DateOfBirth, FirstName, LastName)
	VALUES ('aeinstein30', '5555666777788888', TO_DATE('1960-11-20', 'YYYY-MM-DD'),
'Albert', 'Einstein');

INSERT INTO Individual (Username, PaymentInfo, DateOfBirth, FirstName, LastName)
	VALUES ('tshebs', '1111222233334444', TO_DATE('1900-02-21', 'YYYY-MM-DD'),
'Tommy', 'Shelby');

INSERT INTO Individual (Username, PaymentInfo, DateOfBirth, FirstName, LastName)
	VALUES ('ashebs12', '1111222233334444', TO_DATE('1895-05-10', 'YYYY-MM-DD'),
'Arthur', 'Shelby');

INSERT INTO Individual (Username, PaymentInfo, DateOfBirth, FirstName, LastName)
	VALUES ('pshelby1', '2222333344445555', TO_DATE('1894-10-11', 'YYYY-MM-DD'),
'Polly', 'Shelby');

INSERT INTO Individual (Username, PaymentInfo, DateOfBirth, FirstName, LastName)
	VALUES ('nomoney10', NULL, TO_DATE('1970-11-06', 'YYYY-MM-DD'), 'Dan',
'NoMoney');
```

Insert statements for Organization

```
INSERT INTO Organization (Username, FoundedDate, OrgName)
	VALUES ('wildlife', TO_DATE('2000-08-15', 'YYYY-MM-DD'), 'Wildlife Preservation Inc');

INSERT INTO Organization (Username, FoundedDate, OrgName)
	VALUES ('globewarm', TO_DATE('1950-10-11', 'YYYY-MM-DD'), 'Global Warming is
Real');

INSERT INTO Organization (Username, FoundedDate, OrgName)
	VALUES ('ubcneedsmoney', TO_DATE('2023-10-11', 'YYYY-MM-DD'), 'Give UBC Money
Please');
```

INSERT INTO Organization (Username, FoundedDate, OrgName)
VALUES ('sfuneedsmoney', TO_DATE('2023-10-11', 'YYYY-MM-DD'), 'Give SFU Money Please');

INSERT INTO Organization (Username, FoundedDate, OrgName)
VALUES ('fundbenslife', TO_DATE('2023-10-11', 'YYYY-MM-DD'), 'Ben Wants Money');

Insert statements for Organization_creates_Project

INSERT INTO Organization_creates_Project (ProjectName, OUsername, Description, Balance)
VALUES ('Help Pandas', 'wildlife', 'Pandas are going extinct. Send money.', 0);

INSERT INTO Organization_creates_Project (ProjectName, OUsername, Description, Balance)
VALUES ('Stop Global Warming', 'globewarm', 'Hot in Herre by Nelly', 0);

INSERT INTO Organization_creates_Project (ProjectName, OUsername, Description, Balance)
VALUES ('Open more CPSC110 sections', 'ubcneedsmoney', 'Trust the natural recursion', 0);

INSERT INTO Organization_creates_Project (ProjectName, OUsername, Description, Balance)
VALUES ('Open more CPSC210 sections', 'ubcneedsmoney', 'Java', 0);

INSERT INTO Organization_creates_Project (ProjectName, OUsername, Description, Balance)
VALUES ('Ben's Fund', 'fundbenslife', 'Ben needs money.', 0);

Insert statements for Organization_creates_Post

INSERT INTO Organization_creates_Post(PostID, OUsername, ProjectName, Content, ImageURL, Timestamp)
VALUES (101, 'wildlife', 'Help Pandas', 'Image of Panda', 'https://i.ebayimg.com/images/g/770AAOSwmYFjG-6w/s-l1600.jpg', CURRENT_TIMESTAMP);

INSERT INTO Organization_creates_Post(PostID, OUsername, ProjectName, Content, ImageURL, Timestamp)
VALUES (102, 'globewarm', 'Stop Global Warming', 'Time is running out.', 'https://atmos.earth/wp-content/uploads/2022/10/mm93_evil_corporate_greed_stealing_a_habitable_future_5997b7c6-e742-4ce4-b948-8d22a4abaef4.jpg', CURRENT_TIMESTAMP);

INSERT INTO Organization_creates_Post(PostID, OUsername, ProjectName, Content, ImageURL, Timestamp)
VALUES (103, 'globewarm', 'Stop Global Warming', 'For a green future.', 'https://imageio.forbes.com/specials-images/imageserve/61a7d717a109f9017391549b/c

omparing-green-earth-and-effect-of-air-pollution-from-human-action--glbal-warming/960x0.jpg?height=481&width=711&fit=bounds', CURRENT_TIMESTAMP);

INSERT INTO Organization_creates_Post(PostID, OUsername, ProjectName, Content, ImageURL, Timestamp)
VALUES (104, 'ubcneedsmoney', 'Open more CPSC110 sections', 'Overcrowded CPSC Lecture Hall.', 'https://i.cbc.ca/1.4826104.1537136387!/fileImage/httpImage/image.png_gen/derivatives/16x9_940/lumohacks-hackathon-sfu.png', CURRENT_TIMESTAMP);

INSERT INTO Organization_creates_Post(PostID, OUsername, ProjectName, Content, ImageURL, Timestamp)
VALUES (105, 'fundbenslife', 'Ben's Fund', 'Help me get to class.', 'https://m.media-amazon.com/images/I/71iEjFb5AAL.jpg', CURRENT_TIMESTAMP);

Insert Statements for Account_writes_Comment_on_Post

INSERT INTO Account_writes_Comment_on_Post(CommentID, Username, PostID, Timestamp,Content)
VALUES (0, 'tshebs', 105, CURRENT_TIMESTAMP, 'We're gonna get ye that red bike and we hope to c ye riding it around brum.');

INSERT INTO Account_writes_Comment_on_Post(CommentID, Username, PostID, Timestamp,Content)
VALUES (1, 'cmcdavid1', 104, CURRENT_TIMESTAMP, 'No mean to chirp but thats a crazy');

INSERT INTO Account_writes_Comment_on_Post(CommentID, Username, PostID, Timestamp,Content)
VALUES (2, 'wildlife', 103, CURRENT_TIMESTAMP, 'Glad to see others doing something about the environment');

INSERT INTO Account_writes_Comment_on_Post(CommentID, Username, PostID, Timestamp,Content)
VALUES (3, 'ashebs12', 102, CURRENT_TIMESTAMP, 'No way');

INSERT INTO Account_writes_Comment_on_Post(CommentID, Username, PostID, Timestamp,Content)
VALUES (4, 'ashebs12', 105, CURRENT_TIMESTAMP, 'Ah a nice racing red');

Insert Statements for Individual_makes_Contribution

```
INSERT INTO Individual_makes_Contribution(IUsername, ProjectName, Amount, Timestamp)
        VALUES ('tshebs','Ben's Fund', 7, CURRENT_TIMESTAMP);

INSERT INTO Individual_makes_Contribution(IUsername, ProjectName, Amount, Timestamp)
        VALUES ('ntesla12','Help Pandas', 150, CURRENT_TIMESTAMP);

INSERT INTO Individual_makes_Contribution(IUsername, ProjectName, Amount, Timestamp)
        VALUES ('aeinstein30','Stop Global Warming', 10, CURRENT_TIMESTAMP);

INSERT INTO Individual_makes_Contribution(IUsername, ProjectName, Amount, Timestamp)
        VALUES ('tshebs','Help Pandas', 68, CURRENT_TIMESTAMP);

INSERT INTO Individual_makes_Contribution(IUsername, ProjectName, Amount, Timestamp)
        VALUES ('ashebs12','Open more CPSC110 sections', 100, CURRENT_TIMESTAMP);

INSERT INTO Individual_makes_Contribution(IUsername, ProjectName, Amount, Timestamp)
        VALUES ('ashebs12','Open more CPSC210 sections', 500000,
        CURRENT_TIMESTAMP);
```

Insert Statements for Reward

```
INSERT INTO Reward(RewardName, OUsername)
        VALUES ('T-Shirt', 'wildlife');

INSERT INTO Reward(RewardName, OUsername)
        VALUES ('Water Bottle', 'wildlife');

INSERT INTO Reward(RewardName, OUsername)
        VALUES ('Discount Code', 'globewarm');

INSERT INTO Reward(RewardName, OUsername)
        VALUES ('Tesla Model S', 'ubcneedsmoney');

INSERT INTO Reward(RewardName, OUsername)
        VALUES ('CEO's Favorite Hat', 'fundbenslife');
```

Insert Statements for Individual_receives_Reward

```
INSERT INTO Individual_receives_Reward(RewardName, IUsername)
        VALUES ('CEO's Favorite Hat', 'tshebs');

INSERT INTO Individual_receives_Reward(RewardName, IUsername)
        VALUES ('Water Bottle', 'ntesla12');
```

```
INSERT INTO Individual_receives_Reward(RewardName, IUsername)
        VALUES ('Discount Code', 'aeinstein30');

INSERT INTO Individual_receives_Reward(RewardName, IUsername)
        VALUES ('T-Shirt', 'tshebs');

INSERT INTO Individual_receives_Reward(RewardName, IUsername)
        VALUES ('Tesla Model S', 'ashebs12');
```

Insert Statements for PaymentTier

```
INSERT INTO PaymentTier(PayTierID, ProjectName, OUserName, Description, minAmount,
maxAmount)
        VALUES (0,'Help Pandas', 'wildlife', 'Donate 50$ and receive a t-shirt', 50, 100);

INSERT INTO PaymentTier(PayTierID, ProjectName, OUserName, Description, minAmount,
maxAmount)
        VALUES (1,'Help Pandas', 'wildlife', 'Donate 100$ and receive a water bottle', 100, 250);

INSERT INTO PaymentTier(PayTierID, ProjectName, OUserName, Description, minAmount,
maxAmount)
        VALUES (2,'Stop Global Warming', 'globewarm', 'Donate 10$ and receive a discount
code for water', 10, 10);

INSERT INTO PaymentTier(PayTierID, ProjectName, OUserName, Description, minAmount,
maxAmount)
        VALUES (3,'Open more CPSC210 sections', 'ubcneedsmoney', 'Donate >=100000$ and
receive an electric car', 100000, 1000000);

INSERT INTO PaymentTier(PayTierID, ProjectName, OUserName, Description, minAmount,
maxAmount)
        VALUES (4,'Bens Fund', 'fundbenslife', 'Donate enough money for a subway sandwich
and receive the ultimate reward', 5, 1000000);
```

Insert Statements for PaymentTier_has_Reward

```
INSERT INTO PaymentTier_has_Reward(PayTierID, RewardName)
        VALUES (0, 'T-Shirt');

INSERT INTO PaymentTier_has_Reward(PayTierID, RewardName)
        VALUES (1, 'Water Bottle');
```

```
INSERT INTO PaymentTier_has_Reward(PayTierID, RewardName)
        VALUES (2, 'Discount Code');

INSERT INTO PaymentTier_has_Reward(PayTierID, RewardName)
        VALUES (3, 'Tesla Model S');

INSERT INTO PaymentTier_has_Reward(PayTierID, RewardName)
        VALUES (4, 'CEOs Favorite Hat');
```