TEAM 9

Phase 3

Vamshi Adimulam (vadimulam3)
Corey Caskey (ccaskey6)
Camilo Jaimes (cjaimes3)

Login

- **SELECT * FROM `User` WHERE `Username` = %s**
    - Retrieve user information from database
    - %s is username input

Register

- **INSERT INTO `User`(`Username`, `Password`, `IsAdmin`) VALUES (%s, %s, 0)**
    - Add User to database with input data
    - %s are username input, hashed sequence of password input
- **INSERT INTO `Passenger`(`Username`, `Email`) VALUES (%s, %s)**
    - Add Passenger to database with input data
    - %s is username input, email input
- **INSERT INTO `Breezecard`(`BreezecardNum`, `Value`, `BelongsTo`) VALUES (%s, 0, %s)**
    - Add Breezecard to database from Breezecard input (“Has Breezecard” option)
    - %s are breezecard inputs, username input
- **SELECT * FROM `Breezecard` WHERE `BreezecardNum` = %s AND `BelongsTo` IS NULL**
    - Check if Breezecard owner is NULL (has no owner)
    - %s is breezecard input
- **UPDATE `Breezecard` SET `BelongsTo` = %s WHERE `BreezecardNum` = %s**
    - Give Breezecard that has no owner to username
    - %s are username input, breezecard input
- **INSERT INTO `Conflict`(`Username`, `BreezecardNum`, `DateTime`) VALUES (%s, %s, %s)**
    - Creates a new entry when two passengers claim the same breezecard no.
    - Then the breezecard becomes suspended

Generate New Breezecard (Helper Method)

- **INSERT INTO `Breezecard`(`BreezecardNum`, `Value`, `BelongsTo`) VALUES (%s, 0, %s)**
    - Checks if newly generate card is already in database
    - %s are internally generated Breezecard number, username input

Station Management

- **SELECT * FROM `Station`**
    - Pulls all the information from the Station database
    - Four columns: StopId, Name, EnterFare, ClosedStatus, IsTrain
- **INSERT INTO `Station` VALUES (%s, %s, %s, %s, %s)**
    - Creates a new Station Entry
    - Values: StopId, Name, EnterFare, ClosedStatus, IsTrain
- **INSERT INTO `BusStationIntersection` VALUES (%s, %s)**
    - If the Station is a bus station, it creates a bus station in the BusStationintersection table
    - Values: StopId and Intersection
- **SELECT * FROM `Station` ORDER BY `Name` ASC**

    - Orders all of the stations alphabetically by Name
- **SELECT * FROM `Station` ORDER BY `Name` DESC**
    - Orders all of the stations in reverse alphabetical by Name
- **SELECT * FROM `Station` ORDER BY `StopID` ASC**
    - Orders all of the stations alphabetically by StopId
- **SELECT * FROM `Station` ORDER BY `StopID` DESC**
    - Orders all of the stations in reverse alphabetical by StopId
- **SELECT * FROM `Station` ORDER BY `EnterFare` ASC**
    - Orders all of the stations by increasing fare amount
- **SELECT * FROM `Station` ORDER BY `EnterFare` DESC**
    - Orders all of the stations by decrease fare amount
- **SELECT * FROM `Station` ORDER BY `ClosedStatus` ASC**
    - Orders all of the stations first the ones that Open first then Closed
- **SELECT * FROM `Station` ORDER BY `ClosedStatus` DESC**
    - Orders all of the stations first the ones that are Closed then the ones that are Closed

View Station

- **SELECT * FROM `Station` WHERE `StopID` = %s**
    - Once you select a specific station, you are able to view all of the information about a specific station
    - You get all of the information: StopId, Name, EnterFare, IsClosed, IsTrain
    - What you need is the StopId of the station, the user is trying to access
- **SELECT `Intersection` FROM `BusStationIntersection` WHERE `StopId` = %s**
    - If we know the current station is a bus station, we look at the BusStatioIntersection table
    - Once you know the StopId, you pull the intersection value
- **UPDATE `Station`SET `EnterFare` = %s, `ClosedStatus` = %s WHERE `StopID` = %s**
    - When a user selects a certain station, you can update the fare and status of the station
    - You need the fare price and status of the station
    - You will need the new fare price, new status, and current stop id
- **UPDATE `Station`SET `ClosedStatus` = %s WHERE `StopID` = %s**
    - If the user doesn't want to change the fare price and just the status, this statement updates the row to the new status
    - You will need the new status and current stop id

Get Suspended Cards (Helper Method)

- **SELECT `BreezecardNum`, `Username`, `DateTime`, `BelongsTo` FROM `Conflict` AS C NATURAL JOIN `Breezecard` AS B WHERE C.`BreezecardNum` = B.`BreezecardNum` ORDER BY `BreezecardNum`**
    - Gets result table for suspended cards on BreezecardNum

Suspended

- **SELECT `BelongsTo`, COUNT(*) FROM `Breezecard` WHERE `BelongsTo` = %s GROUP BY `BelongsTo`**

    - Get number of Breezecards for username (prevOwner)
    - %s is username value stored in prevOwner variable
- **SELECT * FROM `Trip` WHERE `BreezecardNum` = %s AND `EndsAt` IS NULL**
    - Check if Breezecard currently in trip
    - %s is Breezecard number stored in cardNumber variable
- **UPDATE `Breezecard` SET `BelongsTo` = %s WHERE `BelongsTo` = %s AND `BreezecardNum` = %s**
    - Give Breezecard to username stored in newOwner variable
    - %s are username stored in newOwner variable, username stored in prevOwner variable, and Breezecard number stored in cardNumber variable
- **DELETE FROM `Conflict` WHERE `BreezecardNum` = %s**
    - Remove conflict instances (unsuspend Breezecard)
    - %s is Breezecard number stored in cardNumber variable

Get Passenger Breezecards (Helper Method)

- **SELECT `BreezecardNum`, `Value` FROM `Breezecard` WHERE `BelongsTo` = %s AND `BreezecardNum` NOT IN (SELECT DISTINCT `BreezecardNum` FROM `Conflict`)**
    - Retrieve active (not suspended) Breezecards for current user
    - %s is username for current user

Pass Breeze cards

- **SELECT `BelongsTo`, COUNT(*) FROM `Breezecard` WHERE `BelongsTo` = %s GROUP BY `BelongsTo`**
    - Get number of Breezecards for current user
    - %s is username for current user
- **SELECT * FROM `Trip` WHERE `BreezeCardNum` = %s AND `EndsAt` IS NULL**
    - Check if Breezecard is currently in a trip
    - %s is Breezecard number stored in cardNumber variable
- **UPDATE `Breezecard` SET `BelongsTo` = NULL WHERE `BreezecardNum` = %s**
    - Remove Breezecard for current user
    - %s is Breezecard number stored in cardNumber variable
- **SELECT * FROM `Breezecard` WHERE `BreezecardNum` = %s**
    - Check if added Breezecard already exists in database
    - %s is Breezecard number stored in cardNumber variable
- **UPDATE `Breezecard` SET `BelongsTo` = %s, `Value` = 0 WHERE `BreezecardNum` = %s**
    - Give Breezecard to username when card owner is NULL
    - %s are username for current user, Breezecard number stored in cardNumber variable
- **INSERT INTO `Conflict`(`Username`, `BreezecardNum`, `DateTime`) VALUES (%s, %s, %s)**
    - Creates new conflict instance (suspends Breezecard)
    - %s are username of current user, Breezecard number stored in cardNumber variable, timestamp of conflict in dateTime variable

- **INSERT INTO `Breezecard`(`BreezecardNum`, `Value`, `BelongsTo`) VALUES (%s, 0, %s)""", [cardNumber, username]**
    - Breezecard doesn't exist so insert it into database
    - %s are Breezecard number stored in cardNumber variable, username of current user
- **SELECT * FROM `Breezecard` WHERE `BreezecardNum` = %s AND `BelongsTo` = %s AND `BreezecardNum` NOT IN (SELECT DISTINCT `BreezecardNum` FROM `Conflict`)**
    - Get active Breezecards for current user
    - %s are Breezecard number stored in breezecard variable, username of current user
- **UPDATE `Breezecard` SET `Value` = %s WHERE `BreezecardNum` = %s**
    - Give Breezecard new money value
    - %s are new value stored in total variable, Breezecard number stored in breezecard variable

Get Trips (Helper Method)

- **SELECT `BreezecardNum` FROM `Breezecard` WHERE `BelongsTo` = %s AND `BreezecardNum` NOT IN (SELECT DISTINCT `BreezecardNum` FROM `Conflict`)**
    - Get all active Breezecards for current user
    - %s is username for current user
- **SELECT `StartTime`, `StartsAt`, `EndsAt`, `Tripfare`, `BreezecardNum` FROM `Trip` WHERE `BreezecardNum` = %s AND `EndsAt` IS NOT NULL**
    - Get all trips for Breezecard number
    - %s is each Breezecard owned by current user

View Trips

- **SELECT `StartTime`,`StartsAt`,`EndsAt`,`Tripfare`,`BreezecardNum` FROM Trip WHERE `BreezecardNum` IN (SELECT `BreezecardNum` FROM Breezecard WHERE `BelongsTo` = %s AND `BreezecardNum` NOT IN (SELECT DISTINCT `BreezecardNum` FROM Conflict)) AND `StartTime` >= %s AND `StartTime` <= %s AND `EndsAt` IS NOT NULL""", [username, start, end])**
    - Get all trips for Breezecards belonging to current user and having a start time falling between the intervals
    - %s are username for current user, lower bound time filter, upper bound time filter

Get Start Stations (Helper Method)

- **SELECT * FROM `Station`**
    - Get all stations
- **SELECT `Intersection` FROM `BusStationIntersection` WHERE `StopID` = %s**
    - Get intersection for specific station stop
    - %s is station stopID

Get End Stations (Helper Method)

- **SELECT `BreezecardNum` FROM `Breezecard` WHERE `BelongsTo` = %s**
    - Get all Breezecards belonging to current user
    - %s is username for current user
- **SELECT `StartsAt`, `BreezecardNum` FROM `Trip` WHERE `BreezecardNum` = %s AND `EndsAt` IS NULL**
    - Get ongoing trip for current user's Breezecards
    - %s is Breezecard number stored in cardNum variable
- **SELECT `IsTrain` FROM `Station` WHERE `StopID` = %s**
    - Get type of station for station stop
    - %s is stopID of starting station
- **SELECT * FROM `Station` where `IsTrain` = %s**
    - Get all stations that have same type of station as starting station
    - %s is type status of starting station
- **SELECT `Intersection` FROM `BusStationIntersection` WHERE `StopID` = %s""", [stopID])**
    - Get intersection for station if it is a bus station
    - %s is stopID of station

Take Trip

- **SELECT `BreezecardNum` FROM `Breezecard` WHERE `BelongsTo` = %s**
    - Get all active Breezecards for current user
    - %s is username for current user
- **SELECT `StartsAt`, `BreezecardNum` FROM `Trip` WHERE `BreezecardNum` = %s AND `EndsAt` IS NULL**
    - Check if any ongoing trips for any of the current user's Breezecards
    - %s is Breezecard number stored in cardnum variable
- **SELECT `IsTrain` FROM `Station` WHERE `StopID` = %s**
    - Get type of stration from stopID
    - %s is stopID of starting station
- **SELECT * FROM `Station` where `IsTrain` = %s**
    - Get all stations that have same type of station as starting station
    - %s is type status of starting station
- **SELECT `Intersection` FROM `BusStationIntersection` WHERE `StopID` = %s**
    - Get intersection for station if it is a bus station
    - %s is stopID of station
- **SELECT * FROM `Station`**
    - Gets all of the information from the Station table
    - Values: StopId, name, EnterFare, ClosedStatus, IsTrain
- **SELECT `Intersection` FROM `BusStationIntersection` WHERE `StopID` = %s**
    - Finds the intersection from the BustStationIntersection table
    - You need the Stop ID to get the information
- **SELECT `BreezecardNum`, `Value` FROM `Breezecard` WHERE `BelongsTo` = %s AND `BreezeCardNum` NOT IN (SELECT DISTINCT `BreezecardNum` FROM `Conflict`)**

    - You are getting the breeze card and the value on that breeze that is not suspended
    - You select a breezecard that is owned by a certain user and you also make sure that the breezecard is not in the Conflict table
    - You need username to calulate
- **UPDATE `Breezecard` SET `Value` = %s WHERE `BreezecardNum` = %s**
    - Update breezecard fare value based on the breezecard number
    - You need the new value and the breezecard number that you wanted to amend
- **INSERT INTO `Trip`(`Tripfare`, `StartTime`, `BreezecardNum`, `StartsAt`, `EndsAt`) VALUES (%s, %s, %s, %s, NULL)**
    - Once a passenger starts their trip, we put the price of the trip, start time, breezecard number, start station
    - The end station is Null until the exit from that station.
    - You need the trip cost, start time, breezecard number, and start station
- **SELECT * FROM `Trip` WHERE `BreezecardNum` = %s AND `EndsAt` IS NULL**
    - Gets all of the information from the Trip table based on the breezecard. This is only the breezecards that are currently in a trip
    - You need: breezecard number
    - You get: TripFare, StartTime, BreezecardNum, StartsAt, EndsAt
- **DELETE FROM `Trip` WHERE `BreezecardNum` = %s AND `EndsAt` IS NULL**
    - Deletes an trip where the passenger has not left the end station
    - Needs to have to avoid a weird bug where it changed start time
    - Need the breezecard number
- **INSERT INTO `Trip`(`Tripfare`, `StartTime`, `BreezecardNum`, `StartsAt`, `EndsAt`) VALUES (%s, %s, %s, %s, %s)**
    - After a passenger leaves Marta, this places their trip into the trip entry
    - You need: Tripfare, StartTime, BreezecardNum, StartsAt, EndsAt

Get Reports

- **SELECT SN.`StopID`, SN.`Name`, IFNULL(ST.`PassIn`,0) AS `PassIn`,IFNULL(ET.`PassOut`,0) AS `PassOut`, (IFNULL(`PassIn`,0) - IFNULL(`PassOut`,0)) AS `Flow`, IFNULL(ST.`SumFare`,0) AS `Revenue`, SN.`IsTrain` FROM (SELECT A.`StopID`, A.`Name`, A.`IsTrain` FROM Station AS A) AS `SN` LEFT OUTER JOIN (SELECT B.`StartsAt`,SUM(B.`Tripfare`) AS `SumFare`,COUNT(*) AS `PassIn` FROM Trip AS B GROUP BY B.`StartsAt`) AS `ST`ON SN.`StopID` = ST.`StartsAt` LEFT OUTER JOIN (SELECT C.`EndsAt`,COUNT(*) AS `PassOut` FROM Trip AS C WHERE C.`EndsAt` IS NOT NULL GROUP BY C.`EndsAt`) AS `ET` ON SN.`StopID` = ET.`EndsAt` ORDER BY `PassIn` DESC, `PassOut` DESC**
    - Get result table from left outer joining Station, Trip, and Trip (flow report columns)

Flow Report

- **SELECT SN.`StopID`, SN.`Name`, IFNULL(ST.`PassIn`, 0) AS `PassIn`, IFNULL(ET.`PassOut`, 0) AS `PassOut` , (IFNULL(`PassIn`, 0) - IFNULL(`PassOut`, 0)) AS `Flow`, IFNULL(ST.`SumFare`, 0) AS `Revenue`, SN.`IsTrain` FROM (SELECT A.`StopID`, A.`Name`, A.`IsTrain` FROM Station AS A) AS `SN` LEFT**

OUTER JOIN (SELECT B.`StartsAt` , SUM(B.`Tripfare`) AS `SumFare` , COUNT(*) AS `PassIn` FROM Trip AS B WHERE `StartTime` >= %s GROUP BY B.`StartsAt`) AS `ST` ON SN.`StopID` = ST.`StartsAt` LEFT OUTER JOIN (SELECT C.`EndsAt` , COUNT(*) AS `PassOut` FROM Trip AS C WHERE C.`EndsAt` IS NOT NULL AND `StartTime` >= %s GROUP BY C.`EndsAt`) AS `ET` ON SN.`StopID` = ET.`EndsAt` ORDER BY `PassIn` DESC, `PassOut` DESC

- # get passengers in, out, flow, and revenue for each station within lower bounded time interval
- Needs the user input for time trip starts at the %s values

- **SELECT SN.`StopID`, SN.`Name`, IFNULL(ST.`PassIn`, 0) AS `PassIn`, IFNULL(ET.`PassOut`, 0) AS `PassOut` , (IFNULL(`PassIn`, 0) - IFNULL(`PassOut`, 0)) AS `Flow`, IFNULL(ST.`SumFare`, 0) AS `Revenue`, SN.`IsTrain` FROM (SELECT A.`StopID`, A.`Name`, A.`IsTrain` FROM Station AS A) AS `SN` LEFT OUTER JOIN (SELECT B.`StartsAt` , SUM(B.`Tripfare`) AS `SumFare` , COUNT(*) AS `PassIn` FROM Trip AS B WHERE `StartTime` <= %s GROUP BY B.`StartsAt`) AS `ST` ON SN.`StopID` = ST.`StartsAt` LEFT OUTER JOIN (SELECT C.`EndsAt` , COUNT(*) AS `PassOut` FROM Trip AS C WHERE C.`EndsAt` IS NOT NULL AND `StartTime` <= %s GROUP BY C.`EndsAt`) AS `ET` ON SN.`StopID` = ET.`EndsAt` ORDER BY `PassIn` DESC, `PassOut` DESC**
  - # get passengers in, out, flow, and revenue for each station within upper bounded time interval
  - Needs the user input for time trip ends for the %s values
- **SELECT SN.`StopID`, SN.`Name`, IFNULL(ST.`PassIn`, 0) AS `PassIn`, IFNULL(ET.`PassOut`, 0) AS `PassOut` , (IFNULL(`PassIn`, 0) - IFNULL(`PassOut`, 0)) AS `Flow`, IFNULL(ST.`SumFare`, 0) AS `Revenue`, SN.`IsTrain` FROM (SELECT A.`StopID`, A.`Name`, A.`IsTrain` FROM Station AS A) AS `SN` LEFT OUTER JOIN (SELECT B.`StartsAt` , SUM(B.`Tripfare`) AS `SumFare` , COUNT(*) AS `PassIn` FROM Trip AS B WHERE `StartTime` >= %s AND `StartTime` <= %s GROUP BY B.`StartsAt`) AS `ST` ON SN.`StopID` = ST.`StartsAt` LEFT OUTER JOIN (SELECT C.`EndsAt` , COUNT(*) AS `PassOut` FROM Trip AS C WHERE C.`EndsAt` IS NOT NULL AND `StartTime` >= %s AND `StartTime` <= %s GROUP BY C.`EndsAt`) AS `ET` ON SN.`StopID` = ET.`EndsAt` ORDER BY `PassIn` DESC, `PassOut` DESC**
  - # get passengers in, out, flow, and revenue for each station within two sided time interval
  - Needs user input of start time and end time for the %s values

Get All Breezecards (Helper Method)

- **SELECT * FROM `Breezecard` WHERE `BreezecardNum` NOT IN (SELECT DISTINCT `BreezecardNum` FROM `Conflict`) GROUP BY `BreezeCardNum` ORDER BY `BreezecardNum` ASC**
  - Get all active Breezecards in the database

Admin Breezecards

- **SELECT * FROM `Breezecard` GROUP BY `BreezeCardNum` ORDER BY `BreezecardNum` ASC**
    - Get all Breezecards for the database (even suspended ones)
- **SELECT * FROM `Conflict` WHERE `BreezecardNum` = %s**
    - Get all conflict instances for Breezecard number
    - %s is Breezecard number stored in data variable
- **SELECT * FROM `Breezecard` WHERE `Value` >= %s AND `Value` <= %s AND `BreezecardNum` NOT IN (SELECT DISTINCT `BreezecardNum` FROM `Conflict`) GROUP BY `BreezeCardNum` ORDER BY `BreezecardNum` ASC**
    - Get all Breezecards within value range and active
    - %s are lower bound value input, upper bound value input
- **SELECT * FROM `Breezecard` WHERE `BreezecardNum` = %s AND `BreezecardNum` NOT IN (SELECT DISTINCT `BreezecardNum` FROM `Conflict`) GROUP BY `BreezeCardNum` ORDER BY `BreezecardNum` ASC**
    - Get Breezecard with number that is active
    - %s is Breezecard number stored in cardNumber variable
- **SELECT * FROM `Breezecard` WHERE `BelongsTo` = %s AND `BreezecardNum` NOT IN (SELECT DISTINCT `BreezecardNum` FROM `Conflict`) GROUP BY `BreezeCardNum` ORDER BY `BreezecardNum` ASC**
    - Get Breezecard with owner and is active
    - %s is username of current user stored in owner variable
- **SELECT * FROM `Breezecard` WHERE `Value` >= %s AND `Value` <= %s GROUP BY `BreezeCardNum` ORDER BY `BreezecardNum` ASC**
    - Get all Breezecards with card value within range
    - %s are lower bound value input, upper bound value input
- **SELECT * FROM `Breezecard` WHERE `BreezecardNum` = %s GROUP BY `BreezeCardNum` ORDER BY `BreezecardNum` ASC**
    - Get all Breezecard information for Breezecard number
    - %s is Breezecard number stored in cardNumber variable
- **SELECT * FROM `Breezecard` WHERE `BelongsTo` = %s GROUP BY `BreezeCardNum` ORDER BY `BreezecardNum` ASC**
    - Get all Breezecards for current user
    - %s is username for current user stored in owner variable
- **"SELECT * FROM `Breezecard` WHERE `BreezecardNum` = %s AND `BelongsTo` = %s AND `BreezecardNum` NOT IN (SELECT DISTINCT `BreezecardNum` FROM `Conflict`) GROUP BY `BreezeCardNum` ORDER BY `BreezecardNum` ASC**
    - Gets all of the information of breezecards that are not suspended
    - It orders the breezecards in ascending order
    - Need: breezecard number and belong username
- **SELECT * FROM `Breezecard` WHERE `BreezecardNum` = %s AND `Value` >= %s AND `Value` <= %s AND `BreezecardNum` NOT IN (SELECT DISTINCT `BreezecardNum` FROM `Conflict`) GROUP BY `BreezeCardNum` ORDER BY `BreezecardNum` ASC**

    - Gets all of the information from the Breezecard table where the value of the breezecard is between two values provided by the user. The other stipulation is the breezecard in not suspended. The data is ordered by ascending breezecards
    - You need: breezecard num, lower bound fare, upper bound fare
    - You get: breezecard num, value, belongsto
- **SELECT * FROM `Breezecard` WHERE `BelongsTo` = %s AND `Value` >= %s AND `Value` <= %s AND `BreezecardNum` NOT IN (SELECT DISTINCT `BreezecardNum` FROM `Conflict`) GROUP BY `BreezeCardNum` ORDER BY `BreezecardNum` ASC**
    - Gets all of the information from the Breezecard table based on who owns them. Stipulation: breezecard is not suspended. It only pulls breezecard whose values are between two user provided values. The data is ordered by ascending beezecards
    - You need: belongs to, lower bound fare, upper bound fare
    - You get: breezecard num, value, belongsto
- **SELECT * FROM `Breezecard` WHERE `BelongsTo` = %s AND `BreezecardNum` = %s GROUP BY `BreezeCardNum` ORDER BY `BreezecardNum` ASC**
    - Gets all of the information from the Breezecard table based on who owns them. The data is ordered by ascending rbeezecards
    - You need: belongs to, breezecard num
    - You get: breezecard num, value, belongsto
- **SELECT * FROM `Breezecard` WHERE `BelongsTo` = %s AND `Value` >= %s AND `Value` <= %s AND `BreezecardNum` = %s AND `BreezecardNum` NOT IN (SELECT DISTINCT `BreezecardNum` FROM `Conflict`) GROUP BY `BreezeCardNum` ORDER BY `BreezecardNum` ASC**
    - Gets all of the information from the Breezecard table based on who owns them and the given breezecard num. Stipulation: breezecard is not suspended. It only pulls breezecard who's values are between two user provided values. The data is ordered by ascending beezecards
    - You need: belongs to, lower bound fare, upper bound fare, breezecard number
    - You get: breezecard num, value, belongsto
- **SELECT * FROM `Breezecard` WHERE `BreezecardNum` = %s AND `Value` >= %s AND `Value` <= %s GROUP BY `BreezeCardNum` ORDER BY `BreezecardNum` ASC**
    - Gets all of the information from the Breezecard table based on who owns them. It only pulls breezecard whose values are between two user provided values. The data is ordered by ascending beezecards
    - You need: belongs to, lower bound fare, upper bound fare
    - You get: breezecard num, value, belongsto
- **SELECT * FROM `Breezecard` WHERE `BelongsTo` = %s AND `Value` >= %s AND `Value` <= %s AND `BreezecardNum` NOT IN (SELECT DISTINCT `BreezecardNum` FROM `Conflict`) GROUP BY `BreezecardNum` ORDER BY `BreezecardNum` ASC**
    - Gets all of the information from the Breezecard table based on who owns them. Stipulation: Breezecard is not suspended. It only pulls breezecard whose values are between two user provided values. The data is ordered by ascending beezecards
    - You need: belongs to, lower bound fare, upper bound fare
    - You get: breezecard num, value, belongsto

- **SELECT * FROM `Breezecard` WHERE `BelongsTo` = %s AND `BreezecardNum` = %s GROUP BY `BreezecardNum` ORDER BY `BreezecardNum` ASC**
    - Gets all of the information from the Breezecard table based on who owns them and breezecard number. The data is ordered by ascending beezecards
    - You need: belongs to, breezecard
    - You get: breezecard num, value, belongsto
- **SELECT * FROM `Breezecard` WHERE `BelongsTo` = %s AND `Value` >= %s AND `Value` <= %s AND `BreezecardNum` = %s AND `BreezecardNum` NOT IN (SELECT DISTINCT `BreezecardNum` FROM `Conflict`) GROUP BY `BreezecardNum` ORDER BY `BreezecardNum` ASC**
    - Gets all of the information from the Breezecard table based on who owns them and breezecard num. Stipulation: breezecard is not suspended It only pulls breezecard whose values are between two user provided values. The data is ordered by ascending beezecards
    - You need: belongs to, lower bound fare, upper bound fare, breezecard number
    - You get: breezecard num, value, belongsto
- **SELECT * FROM `Breezecard` WHERE `BreezecardNum` = %s AND `Value` >= %s AND `Value` <= %s GROUP BY `BreezecardNum` ORDER BY `BreezecardNum` ASC**
    - Gets all of the information from the Breezecard table based on breezecard number. It only pulls breezecard whose values are between two user provided values. The data is ordered by ascending beezecards
    - You need: breezecard number, lower bound fare, upper bound fare
    - You get: breezecard num, value, belongsto
- **SELECT * FROM `Breezecard` WHERE `Value` >= %s AND `Value` <= %s AND `BelongsTo` = %s GROUP BY `BreezecardNum` ORDER BY `BreezecardNum` ASC**
    - Gets all of the information from the Breezecard table based on who owns them. It only pulls breezecard whose values are between two user provided values. The data is ordered by ascending beezecards
    - You need: belongs to, lower bound fare, upper bound fare
    - You get: breezecard num, value, belongsto
- **SELECT * FROM `Breezecard` WHERE `Value` >= %s AND `Value` <= %s AND `BelongsTo` = %s AND `BreezecardNum` = %s GROUP BY `BreezecardNum` ORDER BY `BreezecardNum` ASC""", [lowerValue, upperValue, owner, cardNumber]**
    - Gets all of the information from the Breezecard table based on who owns them. It only pulls breezecard whose values are between two user provided values. The data is ordered by ascending beezecards
    - You need: belongs to, lower bound fare, upper bound fare
    - You get: breezecard num, value, belongsto
- **SELECT * FROM `Breezecard` WHERE `BreezecardNum` = %s**
    - Gets all of the information from the Breezecard table based on the breezecard number.
    - You need: breezecard number
    - You get: breezecard num, value, belongsto
- **UPDATE `Breezecard` SET `Value` = %s WHERE `BreezecardNum` = %s**
    - Updates the new value on the breezecard

    - Need: breezecard number
- **SELECT * FROM `Breezecard` WHERE `BreezecardNum` = %s**
    - Gets all information from breezecard based on the breezecard number
    - You need: breeze card number
    - You get: BreezecardNum, Value, BelongsTo
- **SELECT * FROM `User` WHERE `Username` = %s AND `IsAdmin` = 0**
    - Gets all information from User table based on the username and if they are not an admin
    - Gets: Username, password, IsAdmin
- **UPDATE `Breezecard` SET `BelongsTo` = %s WHERE `BreezecardNum` = %s**
    - Reassign breezecard to new user
    - Needs the new owner and breezecardnum
- **SELECT `BelongsTo`, COUNT(*) FROM `Breezecard` WHERE `BelongsTo` = %s GROUP BY `BelongsTo`**
    - Gets the number of breezecard a all passengers
    - Need: username
- **SELECT * FROM `Trip` WHERE `BreezecardNum` = %s AND `EndsAt` IS NULL**
    - Gets trip information based on breezecard number and is still in their current trip
    - Gets: TripFare, StartTime, BreezecardNum, StartsAt, EndsAt
    - Need: breezecard number
- **UPDATE `Breezecard` SET `BelongsTo` = %s, Value = 0 WHERE `BelongsTo` = %s AND `BreezecardNum` = %s**
    - Reassign breezecard from old owner to new owner
    - You need: new owner, old owner, and the breezecard number
- **DELETE FROM `Conflict` WHERE `BreezecardNum` = %s**
    - Deletes all entries based on the breezecard number
    - Need: BreezecardNum