SQL QUERIES USED:

Log in

```
SELECT *
FROM Manager
WHERE Username = eUsername AND Password = ePassword;

SELECT *
FROM Customer
WHERE Username = eUsername AND Password = ePassword;
```

New User Registration

```
//App reads in eUsername, eEmail, ePassword
//App checks if all fields not filled, username unavailable, password =/= confirm password, email address not valid/unique, error

//Check Username is available (zero)
SELECT *
FROM User
WHERE Username = eUsername;

//Check Email is available (zero)
SELECT *
FROM Customer
WHERE Email = eEmail

//Add to table
INSERT INTO User(Username, Password)
VALUES (eUsername, ePassword);

INSERT INTO Customer(Username, Email)
VALUES(eUsername, eEmail);
```

Add School Info

```
//App checks if email ends in edu
UPDATE Customer
SET isStudent = TRUE;
WHERE Username = eUsername;
```

View Train Schedule

```
//App reads in eTrainNumber
//Check that valid train number entered (nonzero), otherwise, error
SELECT *
FROM TrainRoute
WHERE TrainNumber = eTrainNumber;

//View Train Schedule
SELECT TrainNumber, ArrivalTime, DepartureTime, Name, Location
FROM Stop NATURAL JOIN Station
WHERE TrainNumber = eTrainNumber
ORDER BY ArrivalTime;
```

Make Reservation

```
//User chooses stations for Departs From/Arrives At dropdowns
SELECT Location, Name
FROM Station;
```

Search Train - App allows user to choose eDepartsFrom, eArrivesAt, eDepartureDate

```
CREATE OR REPLACE VIEW GoodArrivals AS
SELECT TrainNumber, ArrivalTime
FROM Stop NATURAL JOIN Station
WHERE Name = eArrivesAt;

CREATE OR REPLACE VIEW GoodDeparts AS
SELECT TrainNumber, DepartureTime
FROM Stop NATURAL JOIN Station
WHERE Name = eDepartsAt;

CREATE OR REPLACE VIEW GoodTrains AS
SELECT TrainNumber, DepartureTime, ArrivalTime
FROM GoodArrivals A NATURAL JOIN GoodDeparts D
WHERE TIMEDIFF(D.DepartureTime, A.ArrivalTime) < 0
GROUP BY TrainNumber;

SELECT TrainNumber, DepartureTime, ArrivalTime,TIMEDIFF(ArrivalTime,
DepartureTime) AS Duration, FirstClassPrice, SecondClassPrice
FROM TrainRoute NATURAL JOIN GoodTrains;
```

//App takes in eNumBags, ePassengerName, eClass [chosen class - “Price”], eClassPrice
//App calculates total cost = (1st/2nd Class Price + 30*(eNumBags-2) [can’t be less than 0]) * [if isStudent == TRUE, 0.8] For each ticket
//Check if student discount applies (t/f)
SELECT IsStudent
FROM Customer
WHERE Username = eUsername;

Dropdown for choosing payment
SELECT CardNumber
FROM PaymentInfo
WHERE Username = eUsername;

Add Card
//App checks all fields are filled, expiration date > current date,
INSERT INTO PaymentInfo(CardNumber, CVV, ExpirationDate, NameonCard, Username)
VALUES(eCardNumber, eCVV, eExpDate, eName, eUsername);

Delete Card - check that there is no active reservation linked to card
SELECT *
FROM Reservation
WHERE eCardNumber = CardNumber AND isCancelled = FALSE;

DELETE FROM PaymentInfo
WHERE CardNumber = eCardNumber;

Check card expiration date is ok
SELECT ExpDate
FROM PaymentInfo
WHERE CardNumber = eCardNumber AND DATEDIFF(CURDATE(), ExpDate) < 0;

//Be sure to increment external counter ReservationID - check what current counter is at
SELECT MAX(ReservationID)
FROM Reservation;

```
//Actually make ticket
INSERT INTO Reservation(ReservationID, isCancelled, Username,
CardNumber)
VALUES(eReservationID, FALSE, eUsername, eCardNumber);

//Actually make ticket - for each ticket under eReservationID
INSERT INTO Reserves(Class, DepartureDate, NumberBags, DepartsFrom,
ArrivesAt, PassengerName, ReservationID, TrainNumber)
VALUES(eClass, eDepartureDate, eNumBags, eDepartsFrom, eArrivesAt,
ePassengerName, eReservationID, eTrainNumber);
```

Update Reservation

```
//App reads in eReservationID, eDepartureDate
//Check if reservation was not made by the customer (zero) OR does not
exist (zero) OR is not already cancelled
SELECT *
FROM Reservation NATURAL JOIN Customer
WHERE ReservationID = eReservationID AND Username = eUsername
AND isCancelled = FALSE;

//Get eTrainNumber, eArrivesAt, eDepartsFrom from Reserves
SELECT TrainNumber,ArrivesAt,DepartsFrom
FROM Reserves
WHERE ReservationID = eReservationID;
```

View Reservations Table

```
CREATE OR REPLACE VIEW ArrivalTime AS
SELECT TrainNumber, ArrivalTime
FROM Stop NATURAL JOIN Station
WHERE TrainNumber = eTrainNumber AND Name = eArrivesAt;

CREATE OR REPLACE VIEW DepartureTime AS
SELECT TrainNumber, DepartureTime
FROM Stop NATURAL JOIN Station
WHERE TrainNumber = eTrainNumber AND Name = eDepartsFrom;

CREATE OR REPLACE VIEW Times AS
SELECT *
FROM ArrivalTime NATURAL JOIN DepartureTime;
```

```
//if FirstClass
SELECT TrainNumber,DepartureDate, DepartureTime, ArrivalTime,
TIMEDIFF(ArrivalTime, DepartureTime) AS Duration, DepartsFrom,
ArrivesAt, Class, FirstClassPrice AS Price, NumberBags, PassengerName
FROM Reserves NATURAL JOIN Times
WHERE ReservationID = eReservationID AND TrainNumber =
eTrainNumber;

//if SecondClass
SELECT TrainNumber,DepartureDate, DepartureTime, ArrivalTime,
TIMEDIFF(ArrivalTime, DepartureTime) AS Duration, DepartsFrom,
ArrivesAt, Class, SecondClassPrice AS Price, NumberBags,
PassengerName
FROM Reserves NATURAL JOIN Times
WHERE ReservationID = eReservationID AND TrainNumber =
eTrainNumber;

//App checks if current date is >1 day before departure
SELECT *
FROM Reserves
WHERE ReservationID = eReservationID AND DATEDIFF(CURDATE(),
DepartureDate) < 1);
```

Get TotalCost to do calculation

```
SELECT TotalCost
FROM Reservation
WHERE ReservationID = eReservationID;
```

Update Total Cost

```
UPDATE Reservation
SET TotalCost = eTotalCost
WHERE ReservationID = eReservationID AND TrainNumber =
eTrainNumber;
```

Update Departure Date

```
UPDATE Reserves
SET DepartureDate = eDepartureDate
WHERE ReservationID = eReservationID AND TrainNumber =
eTrainNumber;;
```

Cancel Reservation

```
//Find reservations made by this user, that are not already cancelled, and can still be cancelled
SELECT Reserves.DepartureDate, Reserves.DepartsFrom, Reserves.ArrivesAt, Reserves.Class, Reservation.ReservationID, Reservation.totalCost
FROM Reservation
INNER JOIN Reserves ON Reservation.ReservationID=Reserves.ReservationID
WHERE Reservation.Username=eUsername AND Reservation.IsCancelled=FALSE AND DATEDIFF(Now(), Reserves.DepartureDate) < -1;

//Find cancellation fee based on date so the customer can confirm
//where eID is equal to the reservation ID of a box reservation by a customer to cancel
SELECT CASE WHEN DATEDIFF(Now(), Reserves.DepartureDate) < -7
THEN (SELECT CASE WHEN (Reservation.totalCost * 0.8) - 50.0 > 0
THEN( Reservation.totalCost * 0.8) - 50.0
ELSE 0 END)
ELSE (SELECT CASE WHEN Reservation.totalCost * 0.5 - 50 > 0
THEN Reservation.totalCost * 0.5 - 50
ELSE 0 END)
END AS `Refund`, Reservation.ReservationID, Reservation.totalCost
FROM Reservation
INNER JOIN Reserves ON Reservation.ReservationID=Reserves.ReservationID
WHERE Reservation.ReservationID=eID;

//After confirming that the user wants to cancel, update the reservation to reflect the cancellation
//executed for each cancellation the customer is making, where eID is the reservation ID of the reservation being cancelled
UPDATE Reservation
SET IsCancelled=TRUE
WHERE ReservationID=eID;

//executed for each cancellation the customer is making, where eRe is the refund for that reservation, and eID is the reservation ID
UPDATE Reservation
```

```
SET totalCost=totalCost-eRe
WHERE ReservationID=eID;
```

View Review

```
//Check that eTrainNumber is valid
SELECT *
FROM Review
WHERE TrainNumber = eTrainNumber;

//App reads in eTrainNumber
SELECT Rating, Comment
FROM Review
WHERE TrainNumber = eTrainNumber;
```

Give Review

```
//App reads in eTrainNumber, eRating, eComment
//Increment ReviewNumbers - check current counter
SELECT MAX(ReviewNum)
FROM Review;

//Check if eTrainNumber is available - else, error
SELECT *
FROM TrainRoute
WHERE TrainNumber = eTrainNumber;

//If eTrainNumber or eRating is empty - error
INSERT INTO Review(Comment, ReviewNum, Rating, Username,
TrainNumber)
Values(eComment, eReviewNum, eRating, eUsername, eTrainNumber);


CREATE OR REPLACE VIEW TopTrains
AS SELECT month(DepartureDate) AS Month, TrainNumber,
Count(R.ReservationID) AS    NumReservations
FROM Reserves INNER JOIN Reservation R
WHERE isCancelled = 0 AND R.ReservationID = Reserves.ReservationID
GROUP BY TrainNumber
LIMIT 3;
```