TRIGGER:

```
CREATE TRIGGER insert_basic_gl
AFTER INSERT ON userData
FOR EACH ROW
BEGIN
  IF NEW.UseBasicGroceryList THEN
    INSERT INTO groceryProduct (UserId, ProductId, Quantity,LocationId, glId) VALUES
(NEW.UserId, 14, 1, 1, 1);
    INSERT INTO groceryProduct (UserId, ProductId, Quantity,LocationId,  glId) VALUES
(NEW.UserId, 180, 1, 1, 1);
    INSERT INTO groceryProduct (UserId, ProductId, Quantity,LocationId,  glId) VALUES
(NEW.UserId, 364, 1, 1, 1);
 END IF;
END;
```

LOCATION CONSTRAINT:

- FOREIGN KEY (UserLocationId) REFERENCES locationData(LocationId)
- This foreign key already enforces the location constraint, so if someone tries to insert a UserLocationId into userData that doesn't exist in locationData, SQL will throw an error

STORED PROCEDURE to add new user:

```
CREATE PROCEDURE AddNewUser (
  IN p_FirstName VARCHAR(255),
  IN p_LastName VARCHAR(255),
  IN p_EmailId VARCHAR(255),
  IN p_PasswordField VARCHAR(255),
  IN p_UserLocationId INT,
  IN p_useBasicGroceryList TINYINT(1)
)
BEGIN
  IF EXISTS (SELECT 1 FROM locationData WHERE LocationId = p_UserLocationId) THEN
    INSERT INTO userData (FirstName, LastName, EmailId, PasswordField, UserLocationId,
UseBasicGroceryList)
    VALUES (p_FirstName, p_LastName, p_EmailId, p_PasswordField, p_UserLocationId,
p_useBasicGroceryList);
  ELSE
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Invalid LocationId: Location does not exist.';
  END IF;
END;
```

STORED PROCEDURE TO SHOW GROCERY LIST W ALL COSTS:

```
CREATE PROCEDURE GetGroceryListWithEnvironmentalCostAndFuel(
  IN inputUserId INT,
  IN inputGL_ID INT
)
BEGIN
  SELECT
    gp.UserId,
    pd.ProductName,
    pd.ProductId,
    ld.locationId AS LocationId,  -- the product's chosen location
    SUM(ec.TotalEmissions * gp.Quantity) AS TotalProductEC,
    AVG(
      5 * (
        6371 * 0.621371 * ACOS(
          COS(RADIANS(ul.Latitude)) * COS(RADIANS(pl.Latitude)) *
          COS(RADIANS(pl.Longitude) - RADIANS(ul.Longitude)) +
          SIN(RADIANS(ul.Latitude)) * SIN(RADIANS(pl.Latitude))
        )
      )
    ) AS EstimatedFuelGallons
  FROM groceryProduct gp
  JOIN productData pd
    ON gp.ProductId = pd.ProductId
  JOIN environmentalCost ec
    ON pd.EC_Id = ec.EC_Id
  JOIN userData u
    ON gp.UserId = u.UserId
  JOIN locationData ul
    ON u.UserLocationId = ul.LocationId
  JOIN locationData pl
    ON gp.LocationId = pl.LocationId
  JOIN locationData ld
```

```
    ON gp.LocationId = ld.LocationId
  WHERE gp.UserId = inputUserId
    AND gp.glId = inputGL_ID
  GROUP BY
    gp.UserId,
    pd.ProductName,
    ld.locationId;
    pd.ProductId;
END;
```

<u>STORED PROCEDURE ➻ DUPLICATES GROCERY LIST:</u>

```
CREATE PROCEDURE CopyGroceryList(
    IN originalListId INT,
    IN inputUserId INT
)
BEGIN
  DECLARE newGlId INT;

  SELECT IFNULL(MAX(gp.glId), 0) + 1 INTO newGlId
  FROM agri.groceryProduct gp
  WHERE gp.userId = inputUserId;

  INSERT INTO agri.groceryProduct (userId, glId, productId, quantity, locationId)
  SELECT
    gp1.userId,
    newGlId,
    gp1.productId,
    gp1.quantity,
    gp1.locationId
  FROM agri.groceryProduct gp1
  JOIN (
    SELECT productId
    FROM agri.groceryProduct
    WHERE glId = originalListId
      AND userId = inputUserId
  ) AS subq
  ON gp1.productId = subq.productId
  WHERE gp1.userId = inputUserId
    AND gp1.glId = originalListId;

END;
```

<u>TRANSACTION 1(in a stored procedure):</u>

```
CREATE PROCEDURE SearchProductsWithDistance(
  IN keyword VARCHAR(255),
  IN userCity VARCHAR(255),
  IN userCountry VARCHAR(255)
)
BEGIN
  DECLARE userLat DECIMAL(8,5);
  DECLARE userLong DECIMAL(8,5);

  SELECT Latitude, Longitude
  INTO userLat, userLong
  FROM locationData
  WHERE
    (City = userCity AND Country = userCountry)
    OR (userCity IS NULL AND Country = userCountry)
  LIMIT 1;

  IF userLat IS NULL OR userLong IS NULL THEN
    SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Invalid user location';
  END IF;

  START TRANSACTION;

  SELECT DISTINCT
    p.ProductName,
    ec.CarbonFootprint_per_kg,
    ec.LandUse_per_kg,
    ec.WaterUse_per_kg,
    ec.TotalEmissions,
    (6371 * acos(
      cos(radians(userLat)) * cos(radians(pl.Latitude)) *
      cos(radians(pl.Longitude) - radians(userLong)) +
      sin(radians(userLat)) * sin(radians(pl.Latitude))
    )) * 0.621371 AS DistanceMiles,
    (
      (6371 * acos(
        cos(radians(userLat)) * cos(radians(pl.Latitude)) *
        cos(radians(pl.Longitude) - radians(userLong)) +
        sin(radians(userLat)) * sin(radians(pl.Latitude))
      )) * 0.621371 * 5
    ) AS FuelUsageGallons
  FROM productData p
  JOIN environmentalCost ec ON p.EC_Id = ec.EC_Id
  JOIN locationData pl ON p.LocationId = pl.LocationId
  WHERE p.ProductName = keyword
     OR p.ProductName LIKE CONCAT('%', keyword, '%');

  COMMIT;
END;
```

TRANSACTION 2(move product from one list to another)(in a stored procedure):

```
CREATE PROCEDURE MoveProductBetweenLists(
  IN inputUserId INT,
  IN inputProductId INT,
  IN sourceListId INT,
  IN targetListId INT
)
BEGIN
  START TRANSACTION;

  -- Insert the product into the target list if it doesn't already exist
  INSERT INTO groceryProduct (UserId, ProductId, Quantity, LocationId, glId)
  SELECT
```

```
    source.UserId,
    source.ProductId,
    source.Quantity,
    source.LocationId,
    targetListId
  FROM groceryProduct AS source
  LEFT JOIN groceryProduct AS target
    ON source.ProductId = target.ProductId
    AND source.LocationId = target.LocationId
    AND target.glId = targetListId
    AND target.UserId = inputUserId
  WHERE source.glId = sourceListId
    AND source.UserId = inputUserId
    AND source.ProductId = inputProductId
    AND target.ProductId IS NULL; -- Only insert if not already exists in target

  -- Delete the product from the source list
  DELETE FROM groceryProduct
  WHERE glId = sourceListId
    AND UserId = inputUserId
    AND ProductId = inputProductId;

  COMMIT;
END;
```