

## Summary

The foundation, TechForAll, solicits technology donations for students across the city. The devices are distributed by criteria determined by the foundation's board of directors. The CompU database tracks the requests made by students to receive a device from the charity and the inventory of devices waiting to be distributed.

TechForAll's mission is to bridge the digital divide and promote equal opportunities in education, TechForAll provides laptops and tablets to students in need, taking into account both their school district and disability.

To achieve their goal, TechForAll partnered closely with local schools and education authorities in the city. They established strong connections with school administrators, teachers, and counselors, who would identify students lacking access to technology due to financial constraints or disabilities.

Through their collaboration, TechForAll has devised a comprehensive process to evaluate each student's needs. They took into consideration the specific school district the student belonged to, ensuring that students from underserved areas were given priority. They also factored in disabilities or special needs, recognizing the importance of providing tailored support to those facing additional challenges.

TechForAll works diligently to gather resources, securing donations from individuals, local businesses, and corporate sponsors who share their vision.

## Goal

The goal of the database is to manage the requests, distribution, and inventory of gifted devices.

## Stakeholders

- Students
- Donors
- TechForAll Foundation

## Business Rules

- A student can request one or more devices
- A district can have more than one school
- A donor can donate more than one device to the inventory
- A district can only have one ranking
- A student can only have one disability ranking
- A district can only have one district rank
- A student can only have one grade rank

## Glossary

*Inventory:* The number of devices available for student requests.

*Students:* The requestors and recipients of devices.

*Requests:* Submitted by students to the foundation to be reviewed by CompU and then approved or denied based on the foundation's criteria.

*Device types:* The brand of laptops or tablets that are donated and distrusted by the foundation.

*Districts:* A special-purpose district that operates local public primary and secondary schools.

*Schools:* An institution for educating students.

*Donors:* The names of people who donated devices to TechForAll.

*District ranks:* Order of districts based on criteria defined by the charity's board of directors. Ranked from 1-5, 1 beginning the 'best.'

*Disability ranks:* Order of disabilities based on criteria defined by the charity's board of directors. Ranked from 1-6. 6 is the most severe.

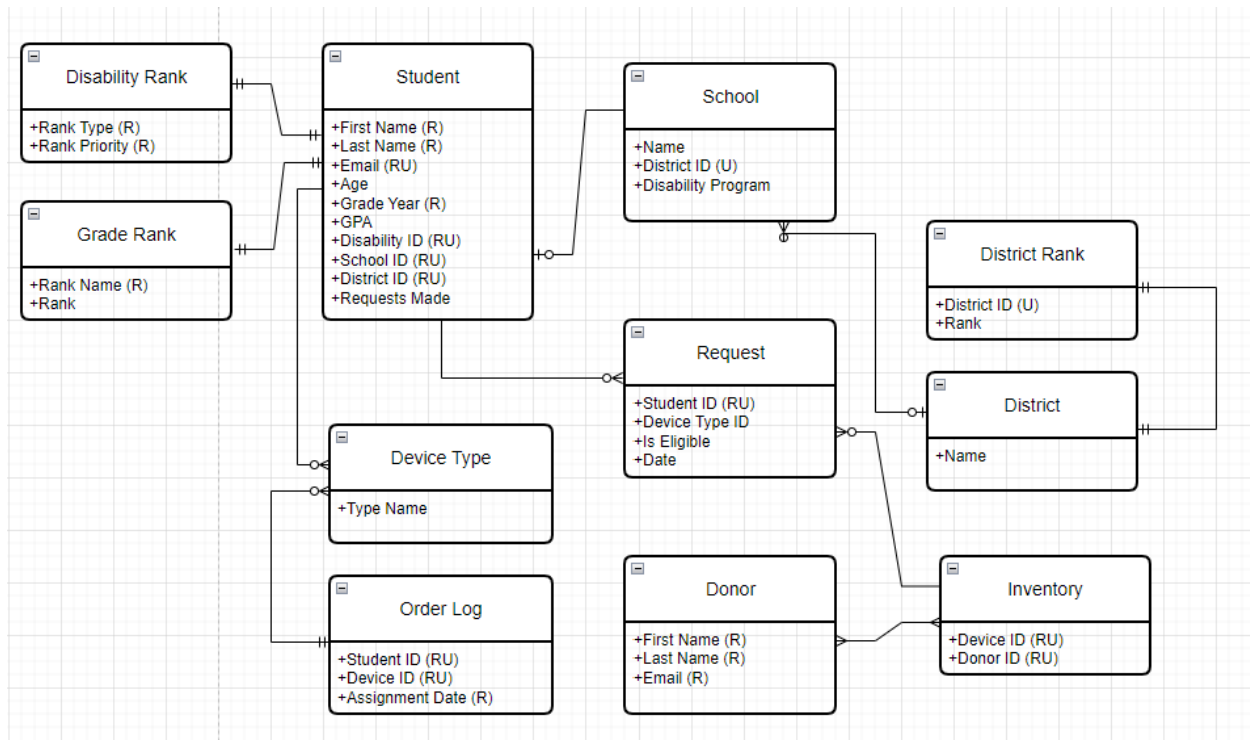
*Grade ranks:* The order of grades from 8<sup>th</sup> grade to 12<sup>th</sup> grade and their rank, 8<sup>th</sup> being 1 and 12<sup>th</sup> being 5.

*Order logs:* The transactions of devices approved for students.

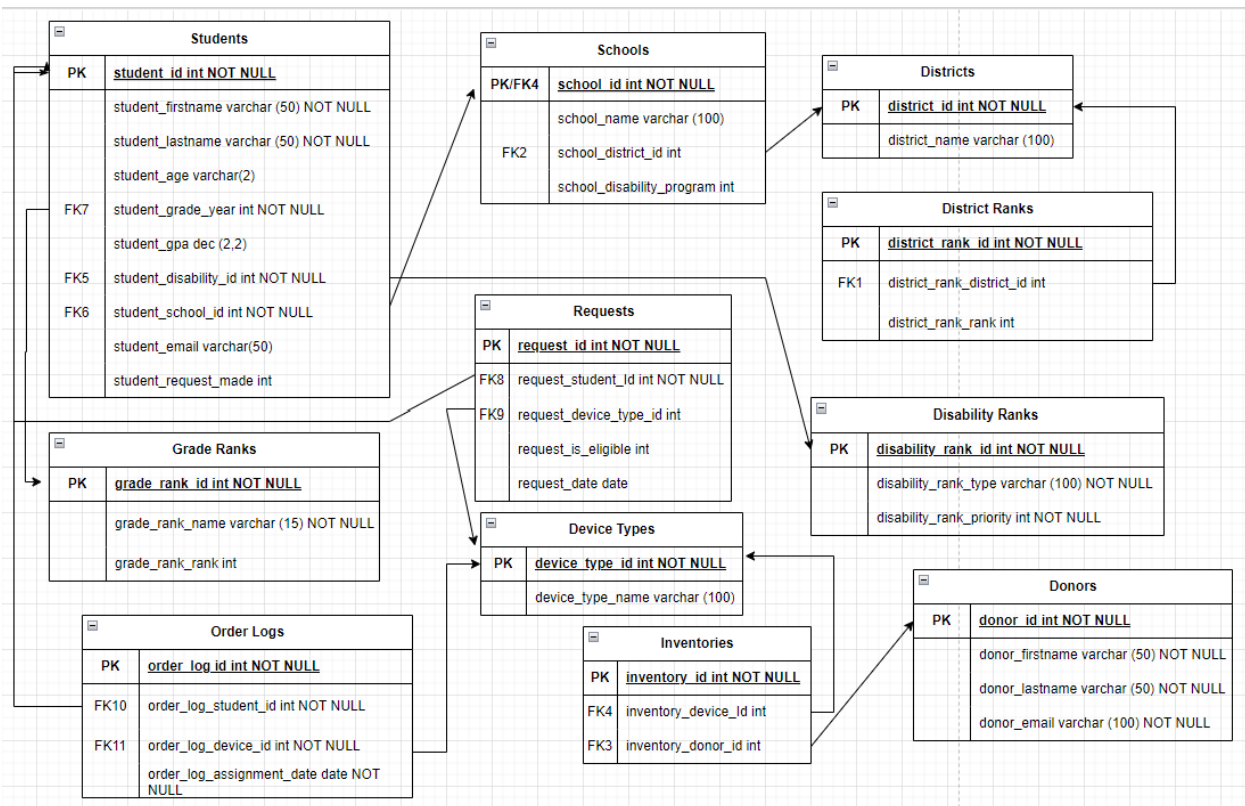
## Data Questions

1. How can we notify a donor once their donated laptops have been given to a student?
2. How many students from each district have requested devices?
3. Who are our top donors?
4. How many students from each district have been approved for device allocation?
5. How many laptops and tablets are currently available in the inventory?
6. How will students be measured against GPA, disability, grade, and district?

## Conceptual Model



## Logical Model



# Physical Database

-- down script

```
if not exists( select * from sys.databases where name = 'compu')
    create database compu
```

GO

use compu

GO

--drop procedures

```
IF OBJECT_ID('EvaluateStudentRequests') IS NOT NULL
```

```
DROP PROCEDURE EvaluateStudentRequests
```

-- Drop the procedure if it already exists

```
IF OBJECT_ID('AssignDeviceToStudent', 'P') IS NOT NULL
```

```
    DROP PROCEDURE AssignDeviceToStudent;
```

GO

-- drop views

```
IF OBJECT_ID('EvaluateStudentRequestView') IS NOT NULL
```

```
    DROP VIEW EvaluateStudentRequestView
```

-- Drop tables in reverse order of creation

```
DROP TABLE IF EXISTS order_logs;
```

```
DROP TABLE IF EXISTS requests;
```

```
DROP TABLE IF EXISTS students;
```

```
DROP TABLE IF EXISTS grade_ranks;
```

```
DROP TABLE IF EXISTS inventorys;
```

```
DROP TABLE IF EXISTS device_types;
```

```
DROP TABLE IF EXISTS donors;
```

```
DROP TABLE IF EXISTS schools;
```

```
DROP TABLE if exists disability_ranks;
```

```
DROP table if exists district_ranks;
```

```
DROP TABLE IF EXISTS districts;
```

-- up script

-- Create table for districts

```
CREATE TABLE districts (  
    district_id INT NOT NULL identity,  
    district_name VARCHAR(100)  
    CONSTRAINT pk_districts_district_id primary key (district_id)  
);
```

-- Insert records for districts

```
INSERT INTO districts (district_name) VALUES ( 'North School District');  
INSERT INTO districts (district_name) VALUES ( 'West School District');  
INSERT INTO districts (district_name) VALUES ('Town Square District');  
INSERT INTO districts (district_name) VALUES ('Uptown District');  
INSERT INTO districts (district_name) VALUES ('East Town District');
```

-- create table for district rank

```
CREATE TABLE district_ranks (  
    district_rank_id INT NOT NULL identity,  
    district_rank_district_id INT,  
    district_rank_rank INT,  
    CONSTRAINT pk_district_ranks_district_rank_id PRIMARY KEY (district_rank_id),  
    FOREIGN KEY (district_rank_district_id) REFERENCES districts(district_id)  
);
```

-- insert into district ranks

```
INSERT INTO district_ranks (district_rank_district_id, district_rank_rank) VALUES ( 1, 5);  
INSERT INTO district_ranks (district_rank_district_id, district_rank_rank) VALUES (2, 3);  
INSERT INTO district_ranks (district_rank_district_id, district_rank_rank) VALUES (3, 2);  
INSERT INTO district_ranks (district_rank_district_id, district_rank_rank) VALUES ( 4, 4);  
INSERT INTO district_ranks (district_rank_district_id, district_rank_rank) VALUES (5, 1);
```

-- create table for disability priority

```
create table disability_ranks(
```

```

disability_rank_id INT NOT NULL identity,
disability_rank_type VARCHAR(100) NOT NULL,
disability_rank_priority INT NOT NULL
CONSTRAINT pk_disability_ranks_disability_rank_id PRIMARY KEY (disability_rank_id)
);

```

```

-- Insert records into disability_ranks

```

```

INSERT INTO disability_ranks (disability_rank_type, disability_rank_priority) VALUES
('Multiple Disabilities', 5);
INSERT INTO disability_ranks (disability_rank_type, disability_rank_priority) VALUES
('Learning Disability', 4);
INSERT INTO disability_ranks (disability_rank_type, disability_rank_priority) VALUES (
'Hearing Disability', 3);
INSERT INTO disability_ranks (disability_rank_type, disability_rank_priority) VALUES ( 'Visual
Disability', 2);
INSERT INTO disability_ranks (disability_rank_type, disability_rank_priority) VALUES (
'Physical Disability', 1);
INSERT INTO disability_ranks (disability_rank_type, disability_rank_priority) VALUES ( 'None',
0);

```

```

-- Create table for schools

```

```

CREATE TABLE schools (
    school_id INT NOT NULL identity,
    school_name VARCHAR(100),
    school_district_id INT,
    school_disabilities_program TINYINT DEFAULT 0,
    CONSTRAINT pk_schools_school_id PRIMARY KEY (school_id),
    CONSTRAINT fk_schools_district
        FOREIGN KEY (school_district_id)
        REFERENCES districts(district_id)
);

```

```

-- Inserting records for District 1

```

```

INSERT INTO schools (school_name, school_district_id, school_disabilities_program)

```



```

VALUES ('North Middle School', 1, 0);
INSERT INTO schools (school_name, school_district_id, school_disabilities_program)
VALUES ('North High School', 1, 1);

-- Inserting records for District 2
INSERT INTO schools (school_name, school_district_id, school_disabilities_program)
VALUES ('West High School', 2, 0);
INSERT INTO schools ( school_name, school_district_id, school_disabilities_program)
VALUES ('West Middle School', 2, 1);
INSERT INTO schools (school_name, school_district_id, school_disabilities_program)
VALUES ('Tom Barady High School', 2, 0);

-- Inserting records for District 3
INSERT INTO schools (school_name, school_district_id, school_disabilities_program)
VALUES ('Town Middle School', 3, 1);
INSERT INTO schools ( school_name, school_district_id, school_disabilities_program)
VALUES ('Town High School', 3, 0);

-- Inserting records for District 4
INSERT INTO schools ( school_name, school_district_id, school_disabilities_program)
VALUES ( 'Uptown Middle School', 4, 1);
INSERT INTO schools ( school_name, school_district_id, school_disabilities_program)
VALUES ( 'Uptown High School', 4, 0);

-- Inserting records for District 5
INSERT INTO schools ( school_name, school_district_id, school_disabilities_program)
VALUES ( 'East Middle School', 5, 0);
INSERT INTO schools ( school_name, school_district_id, school_disabilities_program)
VALUES ( 'East High School', 5, 1);

-- Create table for donors
CREATE TABLE donors (

```

```
donor_id INT NOT NULL identity,  
donor_firstname VARCHAR(50) NOT NULL,  
donor_lastname VARCHAR(50) NOT NULL,  
donor_email VARCHAR(100) NOT NULL  
CONSTRAINT pk_donors_donor_id PRIMARY KEY (donor_id)  
);
```

-- Insert records for donors

```
INSERT INTO donors ( donor_firstname, donor_lastname, donor_email) VALUES ( 'John', 'Smith',  
'john.smith@example.com');  
INSERT INTO donors ( donor_firstname, donor_lastname, donor_email) VALUES ( 'Jane', 'Doe',  
'jane.doe@example.com');  
INSERT INTO donors ( donor_firstname, donor_lastname, donor_email) VALUES ( 'Michael',  
'Johnson', 'michael.johnson@example.com');  
INSERT INTO donors ( donor_firstname, donor_lastname, donor_email) VALUES ( 'Emily', 'Brown',  
'emily.brown@example.com');  
INSERT INTO donors ( donor_firstname, donor_lastname, donor_email) VALUES ( 'David', 'Miller',  
'david.miller@example.com');  
INSERT INTO donors ( donor_firstname, donor_lastname, donor_email) VALUES ( 'Olivia',  
'Taylor', 'olivia.taylor@example.com');  
INSERT INTO donors ( donor_firstname, donor_lastname, donor_email) VALUES ( 'Daniel',  
'Anderson', 'daniel.anderson@example.com');  
INSERT INTO donors ( donor_firstname, donor_lastname, donor_email) VALUES ( 'Sophia', 'Clark',  
'sophia.clark@example.com');  
INSERT INTO donors ( donor_firstname, donor_lastname, donor_email) VALUES ( 'Matthew',  
'Walker', 'matthew.walker@example.com');  
INSERT INTO donors ( donor_firstname, donor_lastname, donor_email) VALUES ( 'Ava', 'Harris',  
'ava.harris@example.com');
```

-- Create table for devices

```
CREATE TABLE device_types (  
    device_type_id INT NOT NULL identity,  
    device_type_name VARCHAR(100)  
    CONSTRAINT pk_device_types_device_type_id primary key (device_type_id)  
);
```

```
-- Insert records into device_types
INSERT INTO device_types ( device_type_name) VALUES ('iPad');
INSERT INTO device_types (device_type_name) VALUES ('Samsung Tablet');
INSERT INTO device_types (device_type_name) VALUES ('Macbook');
INSERT INTO device_types (device_type_name) VALUES ('Chromebook');
```

```
-- Create table for inventory
CREATE TABLE inventorys (
    inventory_id INT NOT NULL identity,
    inventory_device_id INT,
    inventory_donor_id INT,
    CONSTRAINT pk_inventorys_inventory_id primary key (inventory_id),
    CONSTRAINT fk_inventorys_device_id
        FOREIGN KEY (inventory_device_id)
        REFERENCES device_types(device_type_id),
    constraint fk_inventorys_donor_id
        FOREIGN KEY (inventory_donor_id)
        REFERENCES donors(donor_id)
);
```

```
-- Insert records into inventorys
```

```
-- Inserting records for inventory_id 1-10
INSERT INTO inventorys ( inventory_device_id, inventory_donor_id)
VALUES ( 1, 1);
INSERT INTO inventorys ( inventory_device_id, inventory_donor_id)
VALUES ( 2, 2);
INSERT INTO inventorys ( inventory_device_id, inventory_donor_id)
VALUES ( 3, 3);
INSERT INTO inventorys ( inventory_device_id, inventory_donor_id)
VALUES ( 4, 4);
INSERT INTO inventorys ( inventory_device_id, inventory_donor_id)
VALUES ( 3, 5);
INSERT INTO inventorys ( inventory_device_id, inventory_donor_id)
```

```
VALUES ( 1, 6);
INSERT INTO inventories ( inventory_device_id, inventory_donor_id)
VALUES ( 1, 7);
INSERT INTO inventories ( inventory_device_id, inventory_donor_id)
VALUES ( 3, 8);
INSERT INTO inventories ( inventory_device_id, inventory_donor_id)
VALUES ( 4, 9);
INSERT INTO inventories ( inventory_device_id, inventory_donor_id)
VALUES ( 3, 10);
```

```
-- Inserting records for inventory_id 11-20
```

```
INSERT INTO inventories ( inventory_device_id, inventory_donor_id)
VALUES ( 1, 1);
INSERT INTO inventories ( inventory_device_id, inventory_donor_id)
VALUES ( 2, 1);
INSERT INTO inventories ( inventory_device_id, inventory_donor_id)
VALUES ( 3, 1);
INSERT INTO inventories ( inventory_device_id, inventory_donor_id)
VALUES ( 4, 4);
INSERT INTO inventories ( inventory_device_id, inventory_donor_id)
VALUES ( 1, 6);
INSERT INTO inventories ( inventory_device_id, inventory_donor_id)
VALUES ( 1, 6);
INSERT INTO inventories ( inventory_device_id, inventory_donor_id)
VALUES ( 2, 7);
INSERT INTO inventories ( inventory_device_id, inventory_donor_id)
VALUES ( 3, 8);
INSERT INTO inventories ( inventory_device_id, inventory_donor_id)
VALUES ( 4, 6);
INSERT INTO inventories ( inventory_device_id, inventory_donor_id)
VALUES ( 2, 10);
```

```
-- Inserting records for inventory_id 21-30
```

```
INSERT INTO inventories ( inventory_device_id, inventory_donor_id)
VALUES ( 1, 1);
INSERT INTO inventories ( inventory_device_id, inventory_donor_id)
VALUES ( 2, 2);
INSERT INTO inventories ( inventory_device_id, inventory_donor_id)
```

```

VALUES ( 3, 7);
INSERT INTO inventorys ( inventory_device_id, inventory_donor_id)
VALUES ( 4, 7);
INSERT INTO inventorys ( inventory_device_id, inventory_donor_id)
VALUES ( 4, 5);
INSERT INTO inventorys ( inventory_device_id, inventory_donor_id)
VALUES ( 1, 1);
INSERT INTO inventorys ( inventory_device_id, inventory_donor_id)
VALUES ( 2, 1);
INSERT INTO inventorys ( inventory_device_id, inventory_donor_id)
VALUES ( 3, 8);
INSERT INTO inventorys ( inventory_device_id, inventory_donor_id)
VALUES ( 4, 10);
INSERT INTO inventorys ( inventory_device_id, inventory_donor_id)
VALUES ( 3, 10);

```

```
--create table for grade_ranks
```

```

create table grade_ranks(
    grade_rank_id INT NOT NULL ,
    grade_rank_name VARCHAR(15) NOT NULL,
    grade_rank_rank INT NULL
    CONSTRAINT pk_grade_ranks_grade_rank_id PRIMARY KEY (grade_rank_id)
);

```

```
--insert values for grade_ranks
```

```

INSERT INTO grade_ranks (grade_rank_id, grade_rank_name, grade_rank_rank)
VALUES
    (8, '8th Grade', 1),
    (9, '9th Grade', 2),
    (10, '10th Grade', 3),
    (11, '11th Grade', 4),
    (12, '12th Grade', 5);

```

```
-- Create table for students
```

```

CREATE TABLE students (
    student_id INT NOT NULL identity,
    student_firstname VARCHAR(50) NOT NULL,
    student_lastname VARCHAR(50) NOT NULL,
    student_email VARCHAR(50) NOT NULL UNIQUE,
    student_age INT NULL,
    student_grade_year INT NOT NULL,
    student_gpa DECIMAL (3,2) NULL,
    student_disability_id INT not null,
    student_school_id INT NOT NULL,
    student_requests_made INT NULL
    CONSTRAINT pk_students_student_id PRIMARY key (student_id),
    CONSTRAINT fk_students_disability_id
        FOREIGN KEY (student_disability_id)
        REFERENCES disability_ranks(disability_rank_id),
    CONSTRAINT fk_students_school
        FOREIGN KEY (student_school_id)
        REFERENCES schools(school_id),
    constraint fk_students_student_grade_year
        FOREIGN key (student_grade_year)
        REFERENCES grade_ranks(grade_rank_id)
);

```

```

ALTER TABLE students
ALTER COLUMN student_gpa DECIMAL(3, 2);

```

```

ALTER TABLE students
DROP COLUMN student_requests_made;

```

```

-- Insert records for students
INSERT INTO students ( student_firstname, student_lastname, student_email, student_age,
student_grade_year, student_gpa, student_disability_id, student_school_id)
VALUES
    ('John', 'Smith', 'john.smith@example.edu', 16, 10, 3.75, 1, 1),

```

```
( 'Emily', 'Johnson', 'emily.johnson@funnymail.edu', 15, 9, 3.88, 6, 1),
( 'Michael', 'Williams', 'michael.williams@lolmail.edu', 17, 11, 3.60, 4, 1),
( 'Olivia', 'Brown', 'olivia.brown@hilariousmail.edu', 14, 8, 3.92, 5, 2),
( 'James', 'Jones', 'james.jones@crazymail.edu', 15, 9, 2.80, 1, 2),
( 'Sophia', 'Davis', 'sophia.davis@randommail.edu', 16, 10, 3.65, 2, 3),
( 'Daniel', 'Miller', 'daniel.miller@sillymail.edu', 15, 9, 3.95, 1, 4),
( 'Ava', 'Wilson', 'ava.wilson@weirdmail.edu', 17, 11, 3.72, 5, 4),
( 'Liam', 'Taylor', 'liam.taylor@goofymail.edu', 14, 8, 2.81, 1, 5),
( 'Emma', 'Anderson', 'emma.anderson@oddmall.edu', 15, 9, 3.88, 2, 5),
( 'Benjamin', 'Martinez', 'benjamin.martinez@crazymail.edu', 16, 10, 3.70, 4, 6),
( 'Mia', 'Hernandez', 'mia.hernandez@funnymail.edu', 14, 8, 3.91, 5, 6),
( 'Elijah', 'Lopez', 'elijah.lopez@lolmail.edu', 15, 9, 2.76, 5, 6),
( 'Charlotte', 'Garcia', 'charlotte.garcia@hilariousmail.edu', 17, 11, 3.84, 2, 7),
( 'William', 'Smith', 'william.smith@crazymail.edu', 14, 8, 3.90, 6, 7),
( 'Amelia', 'Johnson', 'amelia.johnson@randommail.edu', 15, 9, 3.78, 2, 7),
( 'Alexander', 'Williams', 'alexander.williams@weirdmail.edu', 16, 10, 3.68, 1, 8),
( 'Harper', 'Brown', 'harper.brown@hilariousmail.edu', 14, 8, 3.89, 2, 8),
( 'Daniel', 'Jones', 'daniel.jones@crazymail.edu', 15, 9, 2.83, 6, 9),
( 'Evelyn', 'Davis', 'evelyn.davis@randommail.edu', 17, 11, 3.79, 2, 9),
( 'Jacob', 'Miller', 'jacob.miller@sillymail.edu', 15, 9, 3.82, 6, 9),
( 'Abigail', 'Wilson', 'abigail.wilson@weirdmail.edu', 16, 12, 2.73, 3, 10),
( 'Noah', 'Taylor', 'noah.taylor@goofymail.edu', 14, 8, 3.93, 6, 10),
( 'Sofia', 'Anderson', 'sofia.anderson@oddmall.edu', 15, 9, 3.87, 2, 11),
( 'Logan', 'Martinez', 'logan.martinez@crazymail.edu', 17, 12, 2.77, 5, 11);
```

```
-- Create table for requests
```

```
CREATE TABLE requests (
    request_id INT NOT NULL identity,
    request_student_id INT,
    request_device_type_id INT,
    request_date DATE,
    request_is_eligible TINYINT,
    CONSTRAINT pk_requests_request_id PRIMARY key (request_id),
    CONSTRAINT fk_requests_student
        FOREIGN KEY (request_student_id)
        REFERENCES students(student_id),
    CONSTRAINT fk_requests_device
        FOREIGN KEY (request_device_type_id)
        REFERENCES device_types(device_type_id)
```

```
);
```

```
-- insert into requests
```

```
INSERT INTO requests (request_student_id, request_device_type_id, request_date,  
request_is_eligible)
```

```
VALUES
```

```
(1, 1, '2022-08-15',1),
```

```
(2, 2, '2022-07-04',0),
```

```
(3, 3, '2022-08-17',1),
```

```
(4, 4, '2022-09-19',1),
```

```
(5, 1, '2022-10-25', 0),
```

```
(6, 2, '2022-08-15',1),
```

```
(7, 2, '2022-08-20',0),
```

```
(8, 4, '2022-07-12',1);
```

```
create table order_logs (
```

```
    order_log_id int NOT NULL identity,
```

```
    order_log_student_id int NOT NULL,
```

```
    order_log_device_id int NOT NULL,
```

```
    order_log_assignment_date DATE NOT NULL,
```

```
    CONSTRAINT pk_order_logs_order_log_id primary key (order_log_id),
```

```
    CONSTRAINT fk_order_logs_student_id
```

```
        FOREIGN key (order_log_student_id)
```

```
        REFERENCES students(student_id),
```

```
    CONSTRAINT fk_order_logs_order_log_device_id
```

```
        FOREIGN key (order_log_device_id)
```

```
        REFERENCES device_types(device_type_id)
```

```
)
```

```
-- now lets create a procedure to insert a new request and see if a student who is making a  
request is already in the database
```

```
use compu
```

```
GO
```

```
DROP PROCEDURE IF EXISTS dbo.InsertRequest;
```

```
GO
```

```
CREATE PROCEDURE dbo.InsertRequest
```

```
    @firstName varchar(50),
```

```
    @lastName varchar(50),
```

```
    @email varchar(50),
```



```

    @grade INT,
    @gpa DECIMAL(3,2),
    @disability varchar(100),
    @school varchar(100),
    @request_date DATE,
    @deviceType VARCHAR(100)
AS
BEGIN
    -- checking whether or not the student is a new student in the database
    If Not Exists (Select 1 from students where student_email = @email)
    BEGIN
        --inserting new student
        Insert into students(student_firstname, student_lastname, student_email,
student_grade_year, student_gpa, student_disability_id, student_school_id)
        Values (@firstName, @lastName, @email, @grade, @gpa, (select disability_rank_id from
disability_ranks where disability_rank_type = @disability),
        (select school_id from schools where school_name = @school)
        );
    END
    --insert the request
    INSERT INTO requests (request_student_id, request_device_type_id, request_date)
    VALUES ((select student_id from students where student_email=@email), (select
device_type_id from device_types where device_type_name =@deviceType), @request_date );
END
GO
declare @studentfirstname varchar(50), @studentlastname varchar(50), @email varchar(50),
@grade INT, @gpa decimal (3,2), @disability varchar(100), @school varchar(100),
@request_date DATE, @devicetype VARCHAR(100)
SET @studentfirstname = 'Wade'
set @studentlastname = 'Wilson'
set @email = 'wadewilly@deadpool.edu'
set @grade = 10
set @gpa = 3.82
set @disability = 'Learning Disability'
set @school = 'North High School'
set @request_date = '2022-09-10'
set @devicetype = 'Macbook'
EXEC dbo.InsertRequest @studentfirstname, @studentlastname, @email, @grade, @gpa, @disability,
@school, @request_date, @devicetype
select * from requests

```

```
select * from students
```

	request_id	request_student_id	request_device_type_id	request_date	request_is_eligible
1	1	1	1	2022-08-15	1
2	2	2	2	2022-07-04	0
3	3	3	3	2022-08-17	1
4	4	4	4	2022-09-19	1
5	5	5	1	2022-10-25	0
6	6	6	2	2022-08-15	1
7	7	7	2	2022-08-20	0
8	8	8	4	2022-07-12	1
9	9	26	3	2022-09-10	NULL

	student_id	student_firstname	student_lastname	student_email	student_age	student_grade_year	student_gpa	student_disability_id	student_school_id
17	17	Alexander	Williams	alexander.williams@weirdmail.e..	16	10	3.68	1	8
18	18	Harper	Brown	harper.brown@hilariousmail.edu	14	8	3.89	2	8
19	19	Daniel	Jones	daniel.jones@crazymail.edu	15	9	2.83	6	9
20	20	Evelyn	Davis	evelyn.davis@randommail.edu	17	11	3.79	2	9
21	21	Jacob	Miller	jacob.miller@sillymail.edu	15	9	3.82	6	9
22	22	Abigail	Wilson	abigail.wilson@weirdmail.edu	16	12	2.73	3	10
23	23	Noah	Taylor	noah.taylor@goofymail.edu	14	8	3.93	6	10
24	24	Sofia	Anderson	sofia.anderson@oddmall.edu	15	9	3.87	2	11
25	25	Logan	Martinez	logan.martinez@crazymail.edu	17	12	2.77	5	11
26	26	Wade	Wilson	wadewilly@deadpool.edu	NULL	10	3.82	2	2

-- we can see that a student has been added to the table, students, and a new request as well  
 -- now lets create a view for us to decide if they get a laptop, enter in a student to get a composite score back and see how high that composite score is, anything in the 3 or above is a great score.

GO

```
CREATE VIEW EvaluateStudentRequestView
```

AS

```
SELECT
```

```

s.student_id,
s.student_firstname + ' ' + s.student_lastname as Student_name,
s.student_gpa,
dr.disability_rank_type,
g.grade_rank_name,
r.request_date,
sc.school_name,
dis.district_name,
(
  (s.student_gpa / 4.0) * 0.3 +
  dr.disability_rank_priority * 0.3 +
  g.grade_rank_rank * 0.2 + 1`
  disr.district_rank_rank * 0.2
) AS composite_score,
RANK() OVER (ORDER BY
  (s.student_gpa / 4.0) * 0.3 +
  dr.disability_rank_priority * 0.3 +
  g.grade_rank_rank * 0.2 +
  disr.district_rank_rank * 0.2 DESC
```

```

    ) AS student_rank
FROM
    requests r
    left join students as s on r.request_student_id = s.student_id
    left join disability_ranks as dr on s.student_disability_id = dr.disability_rank_id
    left join schools as sc on s.student_school_id = sc.school_id
    left join grade_ranks as g on g.grade_rank_id = s.student_grade_year
    left join districts as dis on sc.school_district_id = dis.district_id
    left join district_ranks as disr on disr.district_rank_district_id =
dis.district_id
GO
select * from EvaluateStudentRequestView

```

	student_id	Student_name	student_gpa	disability_rank_type	grade_rank_name	request_date	school_name	district_name	composite_score	student_rank
1	1	John Smith	3.75	Multiple Disabilities	10th Grade	2022-08-15	North Middle School	North School District	3.3812500	1
2	5	James Jones	2.80	Multiple Disabilities	9th Grade	2022-10-25	North High School	North School District	3.1100000	2
3	26	Wade Wilson	3.82	Learning Disability	10th Grade	2022-09-10	North High School	North School District	3.0865000	3
4	7	Daniel Miller	3.95	Multiple Disabilities	9th Grade	2022-08-20	West Middle School	West School District	2.7962500	4
5	6	Sophia Davis	3.65	Learning Disability	10th Grade	2022-08-15	West High School	West School District	2.6737500	5
6	3	Michael Williams	3.60	Visual Disability	11th Grade	2022-08-17	North Middle School	North School District	2.6700000	6
7	8	Ava Wilson	3.72	Physical Disability	11th Grade	2022-07-12	West Middle School	West School District	1.9790000	7
8	4	Olivia Brown	3.92	Physical Disability	8th Grade	2022-09-19	North High School	North School District	1.7940000	8
9	2	Emily Johnson	3.88	None	9th Grade	2022-07-04	North Middle School	North School District	1.6910000	9

```

-- Create Procedure for AssignDeviceToStudent if they are evaluated and accepted
GO
CREATE PROCEDURE AssignDeviceToStudent (@student_email VARCHAR(50), @device_type_name
VARCHAR(100))
AS
BEGIN
    -- Check if the student exists in the 'students' table
    IF NOT EXISTS (SELECT 1 FROM students WHERE student_email = @student_email)
    BEGIN
        RAISERROR ('Student does not exist.',12,1)
        RETURN
    END

    -- check if requested device type is available in inventory
    IF NOT EXISTS (Select 1 from inventories left join device_types on device_type_id =
inventory_device_id where device_type_name = @device_type_name)
    BEGIN
        RAISERROR ( 'Device type is not available inventory.', 16, 1)
        RETURN
    END

    -- Check if there is an available device of the requested type

```

```

DECLARE @deviceId INT
SELECT TOP 1 @deviceId = inventory_device_id
FROM inventories left join device_types on device_type_id = inventory_device_id
WHERE device_type_name = @device_type_name

IF @deviceId IS NULL
BEGIN
    RAISERROR('No available device of the requested type.', 12, 1)
    RETURN
END

-- Assign the device to the student
delete from inventories
WHERE inventory_device_id = @deviceId

-- Delete the request
DELETE FROM requests
WHERE request_student_id = (SELECT student_id FROM students WHERE student_email =
@student_Email)
    AND request_device_type_id = (SELECT device_type_id FROM device_types WHERE
device_type_name = @device_type_name)

-- Store the assignment information in a separate table (e.g., assignments_history)
INSERT INTO order_logs (order_log_student_id, order_log_device_id,
order_log_assignment_date)
VALUES ((SELECT student_id FROM students WHERE student_email = @student_Email), @deviceId,
GETDATE())
-- Return success message
SELECT 'Device assigned successfully.' AS Message
END

-- Test Procedure AssignDeviceToStudent
GO

declare @student_email VARCHAR(50) , @device_type_name Varchar(50)
set @student_email = 'emily.johnson@funnyemail.edu'
set @device_type_name = 'Macbook'
EXEC AssignDeviceToStudent @student_email , @device_type_name
select * from order_logs

-- Data Question 1: How can we notify a donor once their donated laptops have been given to a
student?
IF OBJECT_ID('NotifyDonor', 'P') IS NOT NULL

```

```

DROP PROCEDURE NotifyDonor;

GO

CREATE PROCEDURE NotifyDonor (@donor_id INT)
AS
BEGIN
    -- Check if the donor exists in the 'donors' table
    IF EXISTS (SELECT 1 FROM donors WHERE donor_id = @donor_id)
    BEGIN
        -- Get the donor's email
        DECLARE @donor_email VARCHAR(100);
        SELECT @donor_email = donor_email FROM donors WHERE donor_id = @donor_id;

        -- Get the student's full name
        DECLARE @student_fullname VARCHAR(100);
        SELECT @student_fullname = CONCAT(student_firstname, ' ', student_lastname)
        FROM students WHERE student_id = @student_fullname;
    -- Compose the email message
    DECLARE @message VARCHAR(MAX);
    SET @message = CONCAT(
        'Dear Donor,',
        CHAR(50), CHAR(10), CHAR(13), CHAR(10),
        'We want to inform you that your donated device has been assigned to a student.',
        CHAR(50), CHAR(10), CHAR(13), CHAR(10),
        'Student Name: ', @student_fullname,
        CHAR(50), CHAR(10), CHAR(13), CHAR(10),
        'Thank you for your generous donation!',
        CHAR(50), CHAR(10), CHAR(13), CHAR(10),
        'Best regards,',
        CHAR(50), CHAR(10),
        'TechForAll'
    );

    -- Send the email to the donor
    EXEC send_dbmail
        @recipients = @donor_email,
        @subject = 'Device Donation Notification',
        @body = @message;

    PRINT 'Donor notification sent successfully.';
END
ELSE

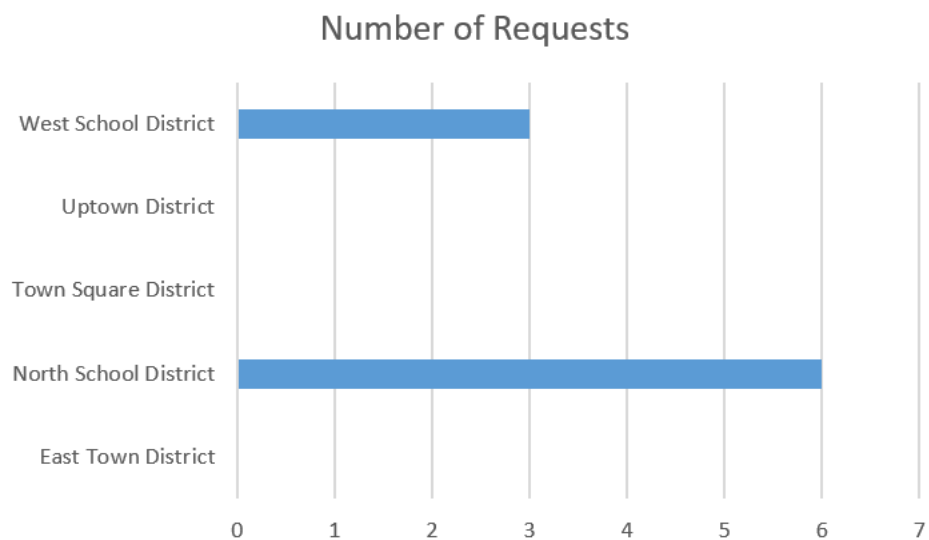
```

```

BEGIN
    PRINT 'Donor does not exist.';
END
END

-- Data Question 2: How many students from each district have requested a device?
SELECT d.district_name , COUNT(r.request_student_id) AS request_count
FROM districts d
LEFT JOIN schools s ON d.district_id = s.school_district_id
LEFT JOIN students st ON s.school_id = st.student_school_id
LEFT JOIN requests r ON st.student_id = r.request_student_id
GROUP BY d.district_name;

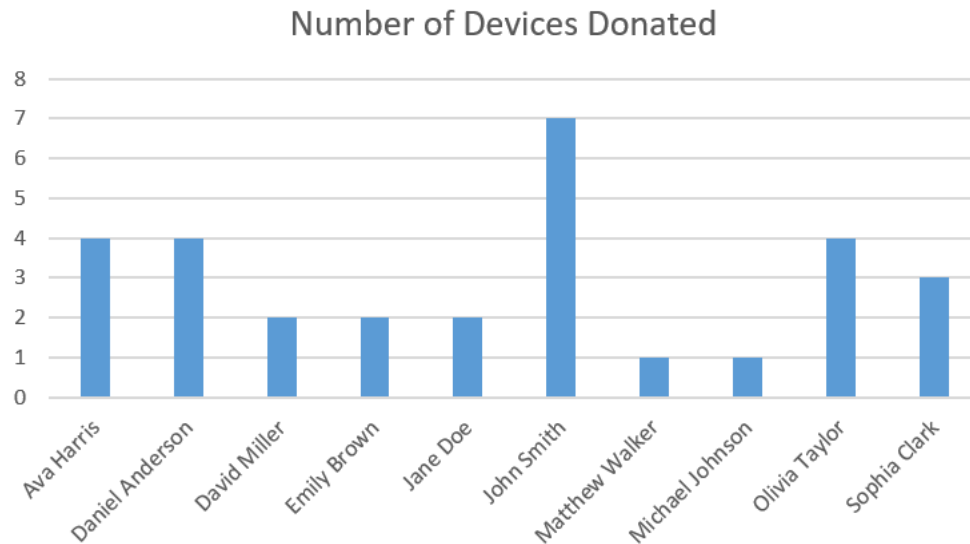
```



```

-- Data Question 3: Who are our top donors?
select d.donor_firstname + ' ' + d.donor_lastname as donor_name,
COUNT(i.inventory_donor_id) as number_of_devices
from donors d
left join inventories as i on d.donor_id = i.inventory_donor_id
group by d.donor_lastname, d.donor_firstname

```



--Data Question 4: How many students from each district have been approved for device allocation?

```
select d.district_name, count(s.student_id) as Number_of_Approved_Students
  from students s
 right join order_logs as o on o.order_log_student_id = s.student_id
 left join schools as sc on s.student_school_id = sc.school_id
 right join districts as d on d.district_id = sc.school_district_id
 group by d.district_name
```

	district_name	Number_of_Approved_Students
1	East Town District	0
2	North School District	1
3	Town Square District	0
4	Uptown District	0
5	West School District	0

--Data Question 5: How many laptops and tablets are currently available in the inventory?

```
select * from inventorys
SELECT
    SUM(CASE WHEN inventory_device_id IN (1,2) THEN 1 ELSE 0 END) AS tablet_count,
    SUM(CASE WHEN inventory_device_id IN (3,4) THEN 1 ELSE 0 END) AS laptop_count
FROM
```

inventorys

	inventory_id ▾	inventory_device_id ▾	inventory_donor_id ▾
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	3	5
6	6	1	6
7	7	1	7
8	8	3	8
9	9	4	9
10	10	3	10
	tablet_count ▾	laptop_count ▾	
1	14	16	

-- Data Question 6: How will students be measured against GPA, disability, grade, and district?

SELECT

```
s.student_id,  
s.student_firstname + ' ' + s.student_lastname as Student_name,  
s.student_gpa,  
dr.disability_rank_type,  
g.grade_rank_name,  
r.request_date,  
sc.school_name,  
dis.district_name,  
(  
  (s.student_gpa / 4.0) * 0.3 +  
  dr.disability_rank_priority * 0.3 +  
  g.grade_rank_rank * 0.2 +  
  disr.district_rank_rank * 0.2  
) AS composite_score,  
RANK() OVER (ORDER BY
```



```

        (s.student_gpa / 4.0) * 0.3 +
        dr.disability_rank_priority * 0.3 +
        g.grade_rank_rank * 0.2 +
        disr.district_rank_rank * 0.2 DESC
    ) AS student_rank
FROM
    requests r
    left join students as s on r.request_student_id = s.student_id
    left join disability_ranks as dr on s.student_disability_id = dr.disability_rank_id
    left join schools as sc on s.student_school_id = sc.school_id
    left join grade_ranks as g on g.grade_rank_id = s.student_grade_year
    left join districts as dis on sc.school_district_id = dis.district_id
    left join district_ranks as disr on disr.district_rank_district_id =
dis.district_id

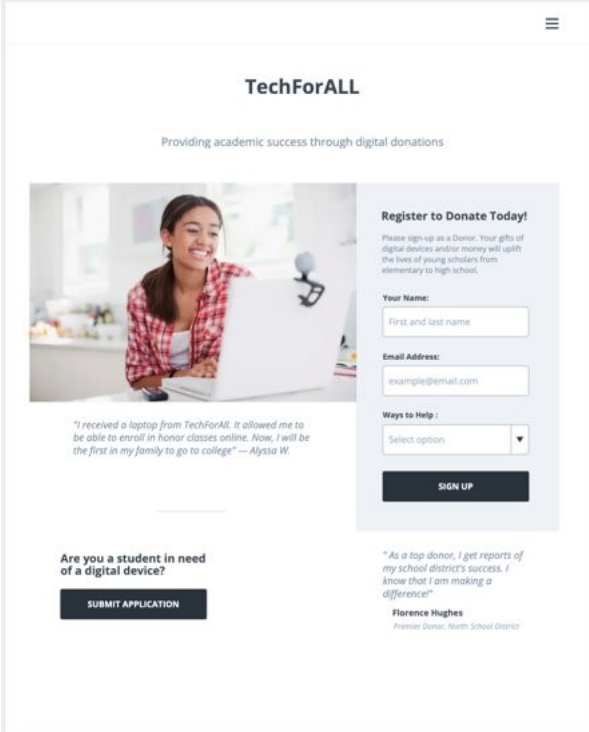
```

composite\_score



- John Smith 3.75 Multiple Disabilities 10th Grade 2022-08-15 North Middle School North School District
- James Jones 2.8 Multiple Disabilities 9th Grade 2022-10-25 North High School North School District
- Wade Wilson 3.82 Learning Disability 10th Grade 2022-09-10 North High School North School District
- Daniel Miller 3.95 Multiple Disabilities 9th Grade 2022-08-20 West Middle School West School District
- Sophia Davis 3.65 Learning Disability 10th Grade 2022-08-15 West High School West School District
- Michael Williams 3.6 Visual Disability 11th Grade 2022-08-17 North Middle School North School District
- Ava Wilson 3.72 Physical Disability 11th Grade 2022-07-12 West Middle School West School District
- Olivia Brown 3.92 Physical Disability 8th Grade 2022-09-19 North High School North School District
- Emily Johnson 3.88 None 9th Grade 2022-07-04 North Middle School North School District


## Application Mock-ups



The mock-up shows a clean, modern website layout. At the top, the 'TechForALL' logo is centered, with a tagline below it. A navigation menu icon is in the top right. The main content area is split into two columns. The left column features a photo of a smiling student, a testimonial quote, and a 'SUBMIT APPLICATION' button. The right column has a 'Register to Donate Today!' section with a form for name and email, a 'WAYS TO HELP' dropdown, and a 'SIGN UP' button. Below this is another testimonial and a 'Premier Donor' credit.

**TechForALL**

Providing academic success through digital donations



*"I received a laptop from TechForAll. It allowed me to be able to enroll in honor classes online. Now, I will be the first in my family to go to college" — Alyssa W.*

**Register to Donate Today!**

Please sign-up as a Donor. Your gifts of digital devices and/or money will uplift the lives of young scholars from elementary to high school.

**Your Name:**

First and last name

**Email Address:**

example@email.com

**Ways to Help :**

Select option

**SIGN UP**

**Are you a student in need of a digital device?**

**SUBMIT APPLICATION**

*"As a top donor, I get reports of my school district's success. I know that I am making a difference!"*

**Florence Hughes**  
Premier Donor, North School District

The TechForAll interface acts as a bridge between the donors and the students that need digital equipment.

Once a donor signs up with their name and email, their profile is created. From their profile, donors are now able to make donations, see their contribution status, as well as observe how well TechForAll is performing.

Students are also able to submit applications for equipment.

[Back](#)

## Student Application

[Student Profile](#)



Disability

District

School

☒ I agree to be the best student I can be

Submit

TechForALL ©

### Thank you for your submission

Stay tune while we review your application.



< Back

## Donor Profile

Home

Profile

Donate

Contacts

Premier Top Donor

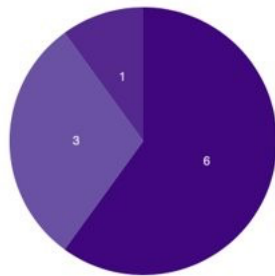


Florence Hughes



Florence.Hughes@outlook.com

Laptops Desktops Tablets



[< Back](#)

## Donation

[Home](#)

[Profile](#)

[Donate](#)

[Contacts](#)

H

Florence Hughes

- ☒ Laptops
- ☐ Desktops
- ☐ Tablets

1 ▾

Qty ▾

Qty ▾

Donate



Thank you for your donation! A student will receive a laptop because of your generosity!!

## Reflection

Throughout this project, we learned the importance of well-kept data and an organized database. The use of the logical model really helped us in shaping the database and making sure we had all the components. If we had to do it again, we think more input data would have been very helpful since it would have made our data questions more useful. Right now, there isn't enough data to make any large statements on which students and from what district are benefiting the most from our charity. None of us were too familiar with Microsoft Access and it proved very difficult to find proper models to showcase what we have done. In the end we think we all learned a valuable amount of SQL and better understand the use of procedures, views, and triggers.

## Summary

Making this database taught us how to use SQL to better understand the use of databases. The charity database we built for TechForALL we feel can be used by an actual company. We really enjoyed this course and what it taught us. The more we worked on the projects and had to figure out roadblocks, the more we got out of this course. Overall, we feel everything we deployed and learned was very useful and has encouraged us to continue learning SQL and better our understanding.