Team 6:
Alper Oner, Astrid Marlina, Isabelle Imacseng, Jeffrey(Jun) Zhang, Jonathan Kung, Peri Yakar, Rick Yang, William Kurniawan

Final Report

Client Description:

SquashDrive is an afterschool program geared towards kids from underprivileged backgrounds based in Oakland (though they sometimes use spaces in the RSF for squash). They provide coaching for squash as well as academic tutoring for children in elementary school through high school. Program enrollment is based on student recommendations from teachers of partner schools. Participants are selected based on fit for academic support and desire to play squash/new sports. There is a trial period for new students gauging their ability to perform in a group setting and receptiveness to academic tutoring. Interviews are finally conducted with the family and students to assess "best fit." Isabelle is a volunteer coach and tutor with the program and her contact is Sean Towers, the Middle School Coordinator who oversees the volunteers. The organization has a little over 100 people, including staff, volunteers, and children.

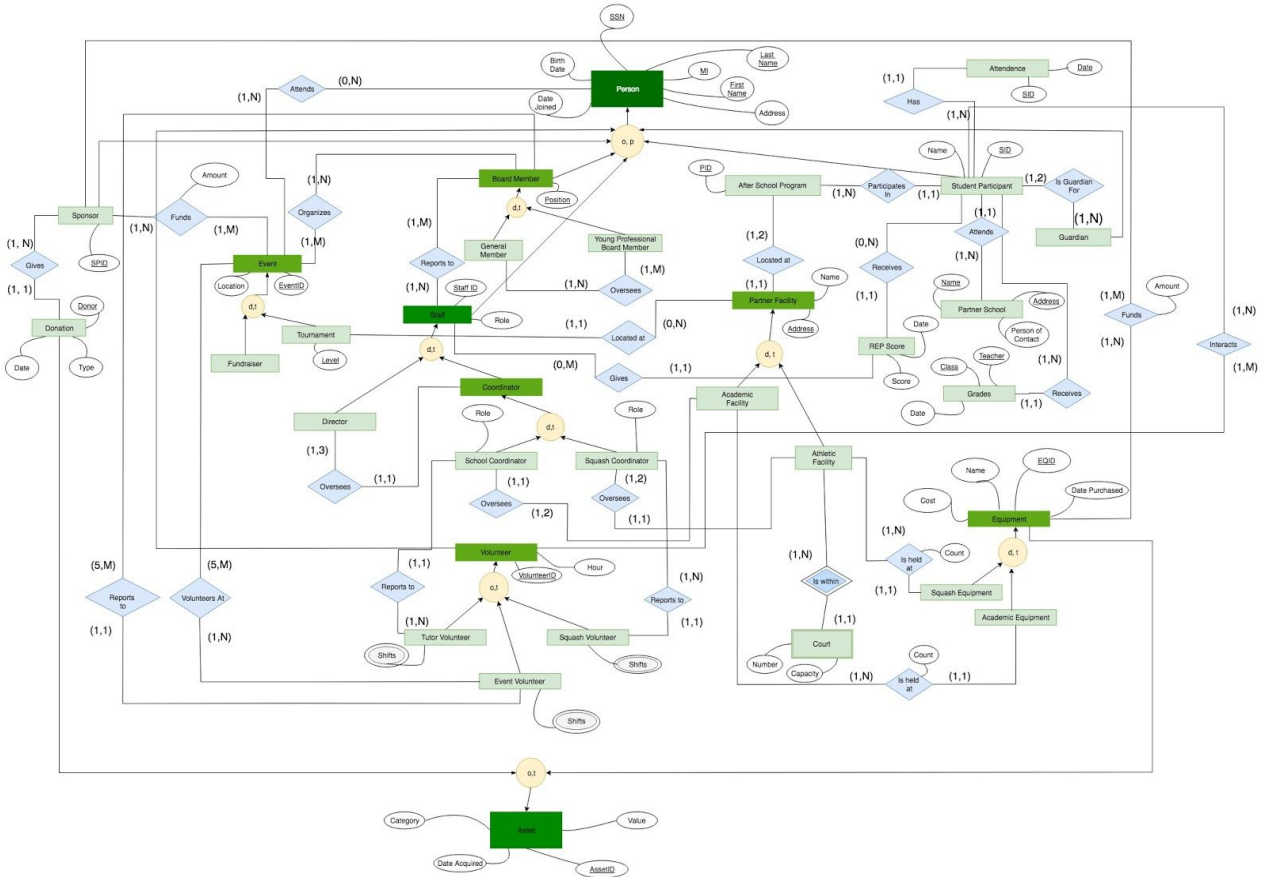
SquashDrive's mission is to help students reach their academic, athletic, and personal potential through individualized educational support, squash support, and character development. The client has no existing database management tool, and their data is currently being recorded on Google Sheets. However, they have the technical fundings that would allow them to use Access if given the prototype. Their current total data size is no greater than 20 KB, which is relatively small.

Project Goals and Methodology:

Our goal for this database is threefold. First, we want to build a database system that can seamlessly accommodate the growth and expansion of new members. Second, the database should be able to efficiently extract specific data on the many students, equipment, facilities, and other recorded data. Finally, the database should understand connections and correlations within the data that can drive operational and programming effectiveness.

To supplement our database, we started by creating mockup EER diagrams and a relational schema that illustrates how different entities in the database are interconnected. In addition, we created 5 useful queries that could extract company insights. We also conducted normalization analysis to illustrate partial dependencies between schemas.

Simplified EER:

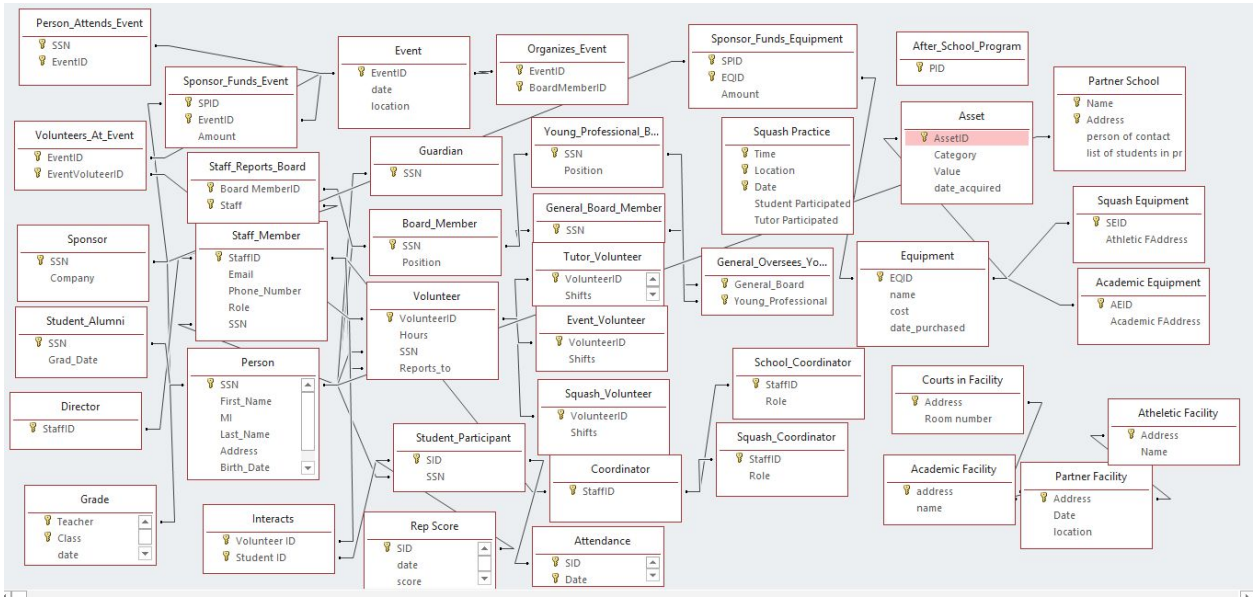


Relational Design Schema:

1. Person (<u>SSN</u>, First_Name, MI, Last_Name, Address, Birth_Date, Date_Joined)
    a. Student_Participant (<u>SID</u>, SSN[1], School[5])
    b. Staff Member (<u>StaffID</u>, email, phone_number, role, SSN[1])
        i. Coordinator (<u>StaffID</u>[1b])
            1. Squash coordinator (<u>StaffID</u>[1b], role)
            2. School coordinator (<u>StaffID</u>[1b], role)
        ii. Director (<u>StaffID</u>[1b])
    c. Guardian (<u>SSN</u>[1])
    d. Volunteer (<u>VolunteerID</u>, Hour, SSN[1], reports_to[1b])
        i. Tutor Volunteer (Shifts, <u>VolunteerID</u>[1d])
        ii. Squash Volunteer (Shifts, <u>VolunteerID</u>[1d])
        iii. Event Volunteer (Shifts, <u>VolunteerID</u>[1d])
    e. Sponsor (company, <u>SSN</u>[1])
    f. Student Alumni (<u>SSN</u>[1], grad_date)

   g. Board Member (SSN[1], Position)
      i. Young Professional Board Member (SSN, Position)
      ii. General Member (SSN)
2. Squash Practice (time, location, date, student participated, volunteer participated)
3. Partner Facility (address, name)
   a. Athletic facility (amount of equipment, address[3])
   b. Academic facility (address[3])
   c. courts in facility (weak entity from building) (room number, address[3])
4. Event (EventID, date, location)
   a. Fundraiser (EventID[4], amount_raised)
   b. Tournament (level, EventID[4], budget)
5. Partner Schools (name, neighborhood, person of contact, list of students in program)
6. Rep score (date, SID[1a], score, given_by[1d])
7. Grade (course_number, semester, SID[1a], teacher_name, grade)
8. Asset (assetID, category, value, date_acquired)
   a. Equipment (EQID[8], name, cost, date_purchased, equipment_type, number_of_years_in_use, condition, value)
      i. Squash Equipments (SEID[8], Athletic Facility[3a])
      ii. Academic Equipments (AEID[8],Academic Facility[3b])
   b. Donation (Donor[1e], date, type, amount)
9. After_School_Program (PID)
10. Attendance (SID[1a], Date)
11. Organizes_Event (EventID[4], Board Member[1g])
12. Volunteers_at_Event (EventID[4], Event_VolunteerID[1dPaiii])
13. Sponsors_Equipment (Sponsor[1e], EQID[8a])
14. SponsorFundsEvent (SPID, EventID[4], Amount, Date_of_Transaction)
15. Interacts (VolunteerID[1di], StudentID[1a])
16. Staff_Reports_Board(Board Member[1g], Staff[1b])
17. General_Oversees_Young (General_Board, Young_Professional)
18. Person_Attends_Event (SSN[1], EventID[4])
19. Sponsor_Funds_Equipment (SPID, EQID[8], amount)

Relational Design - Access Relationship View



Queries:

As part of our analysis, we constructed five SQL queries that would provide insights and value to our client. The queries are as follows:

1. How to best allocate budget between equipment based on condition and value?
2. Which factors are the strongest contributors to participant academic success, and how can we anticipate predict academic success?
3. How do we estimate future donation values and understand donor attribute drivers?
4. How to identify anomalies and compatibilities between student and volunteer interactions?
5. How to forecast facility capacity demand as program enrollment increases?

Query 1:

Question: How to best allocate budget between equipment based on condition and value?

Data extracted: Equipment use-time (current date - date purchased), equipment condition, equipment value

SQL Query: Select EID, Equipment Type, Num_Years, Condition, Value from Equipment

Mathematical Model: Logistic regression model

Detail and Justification:

Under our database design, SquashDrive periodically inspects their equipment inventory and assigns each item a value from 1-10. The value is subjective, but mainly based on the item's usefulness and condition. Items with a value under 7 are typically both useful and in need of repair or replacement. Since inspecting and assigning values to hundreds of pieces of equipment can be time consuming, we built a logistic regression model that predicts whether an item's perceived value is greater than 7. This model could provide value by quickly identifying equipment with lower value scores so that SquashDrive employees don't have to. The equipment predicted to have a value below 7 can then be priorities for the equipment budget.

The logistic regression model is built using the tidyverse package in R. We use the sample() function to split past available data into train and test sets. We then feed our selected variables into a logistic regression model. The model is fit to our training data, and then predicts remaining values for the testing data. The data was randomized with respect to the approximate estimates given to us by Squashdrive.

Tools: R, tidyverse, logistic regression

Data:

	A	B	C	D	E	F	G
1	EID	Equipment_Type	Num_Years	Equipment_Value	Condition	Value_Weight	Value_Over_7
2	1	3	20	9	5	4.5	0
3	2	1	10	3	3.9	1.17	0
4	3	3	7	9	2.7	2.43	0

Logistic Regression Model:

```
Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    -0.830008   0.185958  -4.463 2.84e-05 ***
EID            -0.001028   0.001321  -0.778   0.4390
Equipment_Type  0.130263   0.010834  12.024  < 2e-16 ***
Num_Years      -0.009076   0.017833  -0.509   0.6123
Condition       0.176305   0.095525   1.846   0.0689 .
```

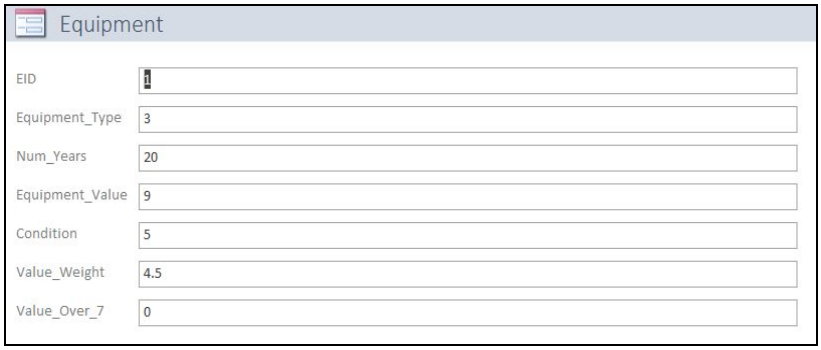
```
       FALSE TRUE
FALSE     14    0
TRUE       1    9
```

R Code:

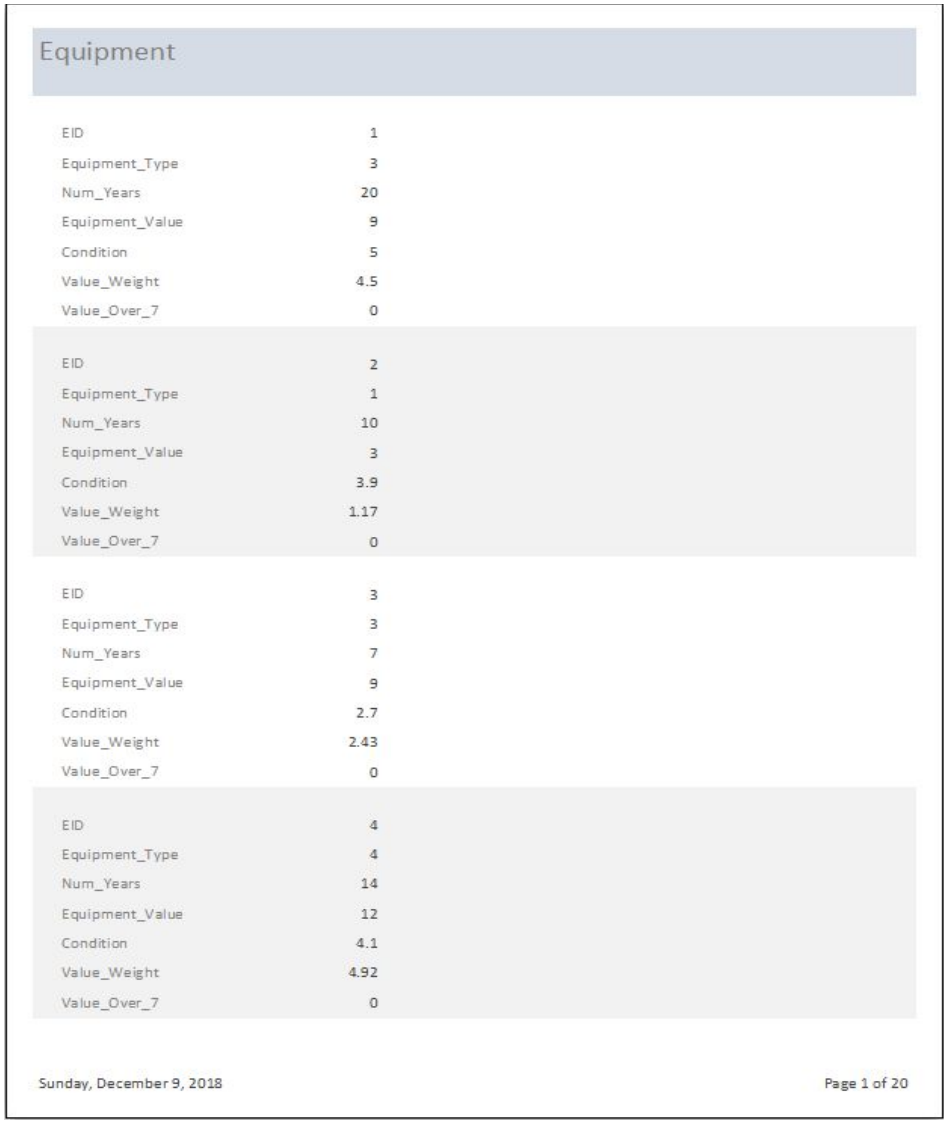
```
#Query 1
setwd("~/Desktop/Year 4 Sem 1/IEOR 115/Design Project")
q1 = read.csv('Query_1.csv')

train.ids1 = sample(nrow(q1), 0.7*nrow(q1))
q1.train = q1[train.ids1,]
q1.test = q1[-train.ids1,]

mod1 = glm(Value_Over_7 ~ . -Equipment_Value -Value_Weight, data = q1)
summary(mod1)

predTestLog <- predict(mod1, newdata=q1.test, type="response")
table(q1.test$Value_Over_7, predTestLog > 0.5)

(14 + 9)/nrow(q1.test)
```

Test Set Accuracy → 95.8%

Form:


Equipment
EID
1
Equipment_Type
3
Num_Years
20
Equipment_Value
9
Condition
5
Value_Weight
4.5
Value_Over_7
0

Report:


Equipment
EID 1
Equipment_Type 3
Num_Years 20
Equipment_Value 9
Condition 5
Value_Weight 4.5
Value_Over_7 0
EID 2
Equipment_Type 1
Num_Years 10
Equipment_Value 3
Condition 3.9
Value_Weight 1.17
Value_Over_7 0
EID 3
Equipment_Type 3
Num_Years 7
Equipment_Value 9
Condition 2.7
Value_Weight 2.43
Value_Over_7 0
EID 4
Equipment_Type 4
Num_Years 14
Equipment_Value 12
Condition 4.1
Value_Weight 4.92
Value_Over_7 0
Sunday, December 9, 2018
Page 1 of 20

Query 2:

Question: Which factors are the strongest contributors to participant academic success, and how can we predict future academic success?

Data extracted: Student Attendance records, REP scores, Academic scores, Student Year, Student Income Level, Tutor Volunteers (who have worked with student), Student Grades, School Attended

SQL Query: Select s.SID, s.Year, s.Income, s.SchoolID, SUM(att.Attended)/ COUNT(att.Date) FROM Student_Participant as s1, Attendance as att WHERE s1.SID = att.SID GROUP BY att.SID; SELECT s2.SID, AVG(g.Grade) FROM Student_Participant as s2, Grade as g WHERE s2.SID = g.GID GROUP BY g.SID; SELECT s3.SID, AVG(rep.Score) from Student_Participant as s3, REP_Score as rep WHERE s3.SID = rep.SID GROUP BY rep.SID);

Average attendance is calculated as the sum of attendance/total days recorded for the student in the program. There exists an attendance record for each student (since their first day)w with att.Attended = 1 if Student attends for that day, 0 otherwise.

Mathematical Model: Multiple Linear Regression/Hypothesis on Regression Parameters:

Details and Justification:

First, we need to extract all possible variables that may impact/correlate with academic success - this includes a variety of factors that are internal and external to the program. All of these will be our potential explanatory variables.

Result from SQL query as exported to Excel (first 28/80 rows) :

SID	Student Year	SchoolID	Student Income	Attendance percentage	Average REP Score	Student Cumulative Grade
1	7	8	18105	0.495	3	1.9
2	11	5	3810	0.11	1	0.4
3	7	10	17374	0.47	3	1.8
4	9	5	23204	0.585	3	2.5
5	11	7	35617	0.975	5	3.9
6	11	6	3740	0.12	1	0.4
7	10	4	19405	0.545	2	2.1
8	8	9	24882	0.625	3	2.7
9	5	2	19328	0.475	3	2
10	10	10	21117	0.55	3	2.3
11	10	1	35712	0.9	5	3.8
12	8	6	17861	0.485	3	1.9
13	5	2	4807	0.135	1	0.5
14	4	3	3753	0.12	1	0.4
15	10	3	3719	0.1	1	0.4
16	6	3	30590	0.785	3	3.3
17	9	6	0	0.02	1	0
18	7	6	34588	0.9	4	3.7
19	7	10	25576	0.65	3	2.7
20	9	5	12107	0.335	2	1.3
21	4	7	35616	0.91	4	3.9
22	6	10	26195	0.695	3	2.8
23	5	8	2924	0.085	0	0.3
24	10	6	31084	0.85	4	3.4
25	9	4	29906	0.77	3	3
26	6	1	34728	0.85	4	3.6
27	4	5	9342	0.25	1	1
28	12	6	9905	0.25	2	1.1

We will create a dependent variable for GPA and use this variable for Multiple Linear Regression $y = \beta_1 + \beta_2 x_2 + \ldots \beta_n x_3 + u$. This will be executed through the Excel Data Analysis Add-In to produce output tables for Interpret Regression Coefficients and Interpret Regression Statistics.

Summary Output from Excel:

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.9993394
R Square	0.9986793
Adjusted R Square	0.9985876
Standard Error	0.0467114
Observations	78

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	5	118.79585	23.75917	10888.916	3.81E-102
Residual	72	0.157101	0.002182		
Total	77	118.95295			

	Coefficients	*tandard Erro*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	-0.042229	0.0287362	-1.469538	0.0146043	-0.099514	0.0150557	-0.099514	0.0150557
Student Year	0.0026043	0.0022665	1.1490491	0.2543386	-0.001914	0.0071224	-0.001914	0.0071224
SchoolID	0.003554	0.0017998	-1.974677	0.0521404	-0.007142	3.381E-05	-0.007142	3.381E-05
Student Income	6.186E-05	5.233E-06	11.821037	1.564E-18	5.143E-05	7.229E-05	5.143E-05	7.229E-05
Attendance percentage	1.651032	0.2236419	7.38248	2.166E-10	1.20521	2.096854	1.20521	2.096854
Average REP Score	0.027656	0.0185457	1.4912332	0.0402687	-0.009314	0.0646261	-0.009314	0.0646261

We test for independent variable null hypotheses and eliminate independent variables with high p-values (keep variables with p-value < 0.05). Apply the final regression equation to predict academic GPA for all student participants.

From our data: p-values are sufficiently low for 4 variables: Average REP Score, Attendance percentage, Student Income, and School ID. This implies that academic performance is linked to all of these variables. We discard Student Year as a potential variable.

Our final equation uses coefficients from the selected variables:

$$y = 0.00355403447052407\, x_1 + 0.0000618617913585998\, x_2 + 1.65103196285203\, x_2 + 0.0276559574178999\, x_3 - 0.0422289744230697$$

This equation can now be applied to incoming students to anticipate academic progress, and track expected progress for students as they continue through the program.

Tools: SQL, Excel Data Analysis Add-in (Regression)

Query 3:

Question: How do we estimate future donation values and understand donor attribute drivers?

Data extracted: Donor ID, Donor income, Donation from Past Year, Donor Gender, Donor Age, Total Donated from Donor (across all charitable donations), Number of Donations

SQL Query: Select d.DonorID, d.income_level, d.past_year_donation, d.gender, d.age, d.Total_Donated, count (past_year_donation) From Sponsor as d Group by DonorID

Mathematical Model: Random Forest

Details and Justification:

Squashdrive should try to understand their donor profiles through data analysis, and use a model based on recorded donor attributes to estimate future donation amounts from particular or prospective donors. This will help them to project out future donation inflows for upcoming years and this projection can be useful for both short and long term planning and budgeting. They can approximately know how much funds that will be available in the upcoming year. Furthermore, they can start planning to allocate the funds into replacing old equipments, creating an event, etc. This model can also be used to understand which attributes are primary drivers for high donation levels, and this insight can help SquashDrive attune their donor acquisition strategy for maximum impact.

The random forest model takes a balanced "wisdom of the crowd" approach by running 500 bootstrapped samples of each donor, making a donation prediction for each sample, and then outputting the mean prediction of the 500 samples. In general, random forests tend to perform well on predictions with a wide range of numerical values, so we expect the model to perform well on this query. The data was randomized with respect to the approximate estimates given to us by Squashdrive.

We started by querying data around existing donors:

Result from SQL query as exported to Excel:

	A	B	C	D	E	F	G	H
1	DonorID	Income level	Current_Year_Do	Past Year Cumu Donation	Past Year Donation	Gender	Age	# of past donations
2	1	763594	2516	6481	2719	1	56	1
3	2	656430	4798	6239	4393	0	44	4
4	3	215599	1916	12792	4140	0	40	1

This data is fed into RStudio:

```
> names(q3)
[1] "DonorID"                    "Income.level"
[3] "Current_Year_Donation_SD"   "Past.Year.Cumu.Donation"
[5] "Past.Year.Donation"         "Gender"
[7] "Age"                        "X..of.past.donations"
```

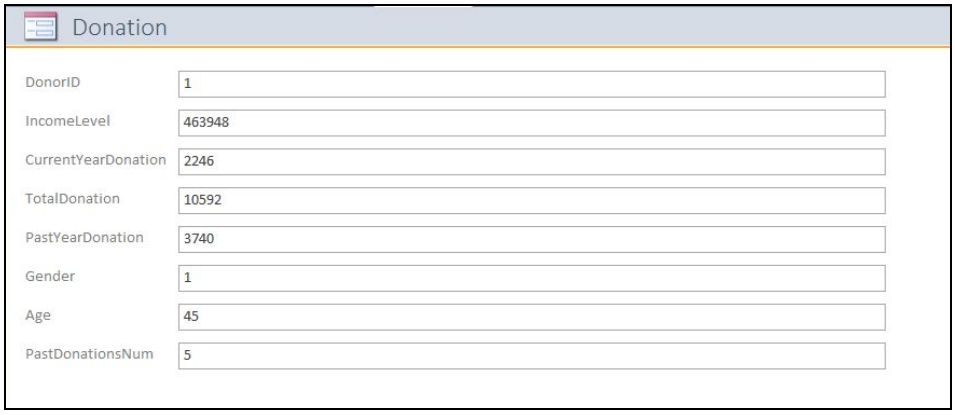
Test Set RMSE → $736.83

R Code:

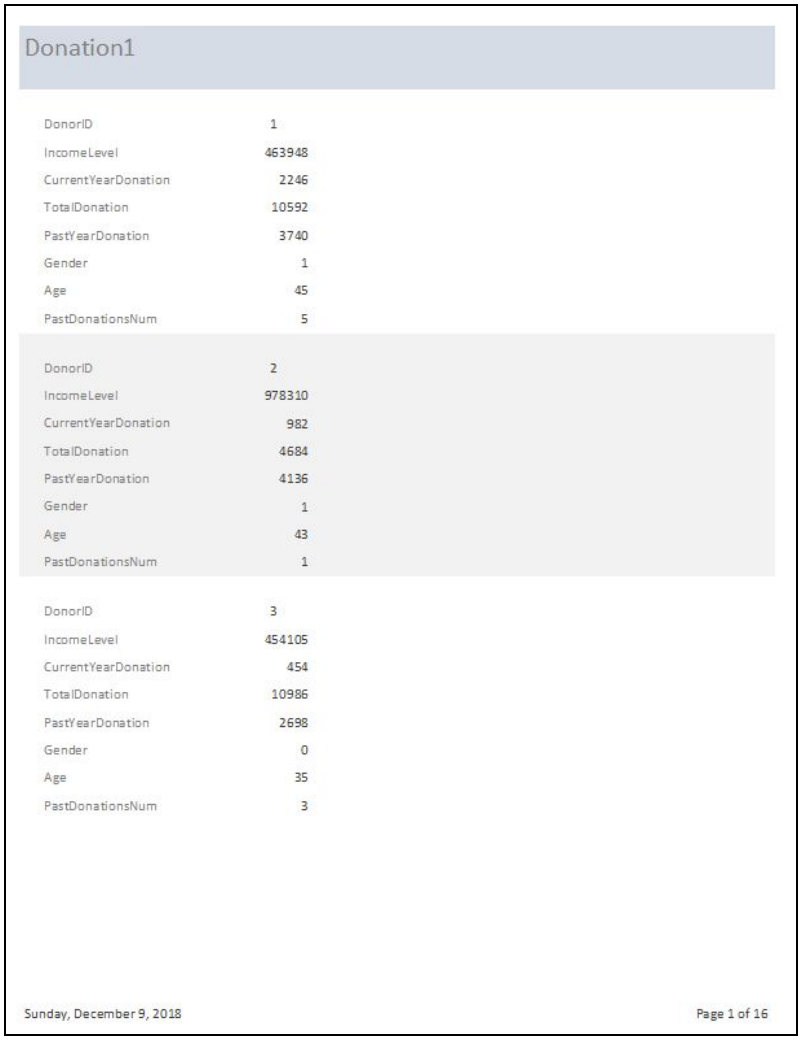
```
#Query 3
q3 = read.csv('Query_3.csv')

train.ids3 = sample(nrow(q3), 0.7*nrow(q3))
q3.train = q3[train.ids3,]
q3.test = q3[-train.ids3,]

q3.rf <- randomForest(Current_Year_Donation_SD ~ .,
                      data = q3.train)

q3.rf$results
summary(q3.rf)
q3pred.rf <- predict(q3.rf, newdata = q3.test)
q3pred.rf

mean(abs(q3pred.rf - q3.test$Current_Year_Donation_SD))
```

Tools: Rstudio, SQL

Form:



Report:



Donation1

DonorID	1
IncomeLevel	463948
CurrentYearDonation	2246
TotalDonation	10592
PastYearDonation	3740
Gender	1
Age	45
PastDonationsNum	5
DonorID	2
IncomeLevel	978310
CurrentYearDonation	982
TotalDonation	4684
PastYearDonation	4136
Gender	1
Age	43
PastDonationsNum	1
DonorID	3
IncomeLevel	454105
CurrentYearDonation	454
TotalDonation	10986
PastYearDonation	2698
Gender	0
Age	35
PastDonationsNum	3

Sunday, December 9, 2018 Page 1 of 16

Alternative Query 4:

Question: How to identify anomalies and compatibilities between student and volunteer interactions?

Data extracted: Student/volunteer interactions, REP scores (both by student and grouped by volunteer who gave REP score), mean of REP scores by student and by student/volunteer pair

SQL Query: We constructed 3 different tables in SQL from the REP score table: average scores grouped by volunteer, student, and both.

SELECT SID, AVG(score) as avg_score FROM Rep_Score GROUP BY SID; SELECT givenBy, AVG(score) as avg_score FROM Rep_Score GROUP BY givenBy; SELECT givenBy, SID, AVG(score) as avg_score FROM Rep_Score GROUP BY SID, givenBy;

Details and Justification:

Group all REP (respect, enthusiasm, preparedness) scores for each day by the volunteer member who assigned the scores and compare those scores using side-by-side boxplots or violin plots. If using SQL, extract a table showing all interactions of volunteers and students with the REP score given for that day. Then group the data by both student and volunteer, aggregated by average REP score.

This will serve several purposes. If one volunteer gives a usually high-scoring student a low REP score consistently, a staff member could investigate and see if the pairing is not compatible or if the student just happens to be having a bad day on the days for which that volunteer is working with that student.

If a volunteer is giving low scores to students, a staff member should check which students the volunteer interacted with and see if the students are ones that are usually more problematic or if the consistently low REP scores are due to the volunteer being overwhelmed or otherwise not properly equipped to interact with the students, in which case they could offer the volunteer more training or in more extreme cases, replace the volunteer. However, could just be that that specific volunteer likes to give lower REP scores. On the other hand, if someone consistently gives very high REP scores, the staff may want to place them with kids who are getting lower REP scores to see if they may be able to better connect with the student.
Tools: seaborn, matplotlib/python

We constructed the synthetic data for this query with the following code:

```
dates = []
for i in np.arange(100):
    y = np.random.randint(2012, 2018)
    m = np.random.randint(1, 12)
    if m in [1, 3, 5, 7, 8, 10, 12]:
        d = np.random.randint(1, 31)
    elif m == 2:
        d = np.random.randint(1, 28)
    else:
        d = np.random.randint(1, 30)
    dates.append(datetime.date(y, m, d))

d5 = {'date': dates,
      'score': np.random.choice([3, 3, 3, 3, 2, 2, 2, 1], 100)}
rep = pd.DataFrame(data = d5)
rep['SID'] = interaction['student']
rep['givenby'] = interaction['volunteer']
```

	date	score	SID	givenby
0	2014-06-04	3	3	3
1	2017-05-12	2	3	2
2	2016-01-29	1	10	1
3	2015-02-19	3	20	1
4	2014-03-25	2	9	6

We decided to use pandas dataframes because it is the easiest to use with Seaborn, a python visualization tool. For the REP Score dataframe, the key attributes are SID, VolunteerID (givenBy), and date. The dates are random dates in the year formatted in python.datetime as (Year, Month, Day). For the actual REP scores, we asked Sarah and Sean what the general breakdown of scores is and randomly assigned the REP scores to each student/volunteer pair from a list with approximate quantities they gave us (approximately: ½ get 3s, and of those who don't, about ¾ get 2s and ¼ get 1s). The SID and givenBy are taken from the Interacts table.

Data in Access from SQL queries:

SID	avg_score
1	2.5
2	2.5
3	2.5
4	1.66666666666667
5	2
6	2.5

givenBy	avg_score
1	1.66666666666667
2	2.33333333333333
3	2.5
4	2.5

givenBy	SID	avg_score
1	3	2
1	4	1
1	6	2
2	1	2
2	3	3
2	5	2
3	1	3
3	2	3
3	4	2
4	2	2

Data in pandas dataframes (different data values due to randomization):

	score
givenby	
1	2.422741
2	2.344371
3	2.426791
4	2.415512
5	2.260355
6	2.408955

	score
SID	
1	2.369369
2	2.387097
3	2.380000
4	2.342857
5	2.405660
6	2.298851
7	2.423423
8	2.504629
9	2.358491

SID	1	2	3	4	5	6	7	8	9	10	11	12
givenby												
1	2.428571	2.545455	2.409091	2.235294	2.437500	2.777778	2.363636	3.000000	2.315789	2.263158	2.434783	2.565217
2	2.150000	2.454545	2.357143	2.428571	2.555556	2.055556	2.533333	2.999999	2.545455	2.500000	2.357143	2.000000
3	2.666667	2.090909	2.428571	2.363636	2.350000	2.214286	2.277778	2.583333	2.583333	2.411765	2.444444	2.642857
4	2.478261	2.458333	2.588235	2.333333	2.285714	2.388889	2.500000	3.000000	2.291667	2.444444	2.642857	2.529412
5	2.000000	2.333333	2.230769	2.444444	2.500000	2.000000	2.444444	2.444444	2.176471	2.272727	2.095238	2.294118
6	2.357143	2.388889	2.285714	2.250000	2.352941	2.571429	2.437500	1.000000	2.391304	2.230769	2.500000	2.458333

```
byBoth = pd.pivot_table(rep, values='score', index='givenby', columns='SID')
```

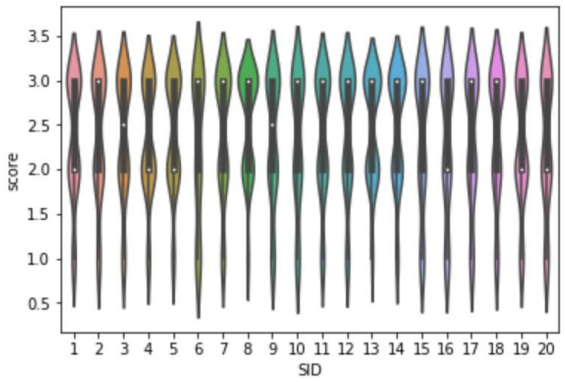
This is a pivot table where all of the values are the average of scores for that student/volunteer pair. In this example, we inspected the relation between student (SID 8) and volunteer

(VolunteerID 6), because that pair has a score of 1 but student 8 got relatively high scores otherwise. From the pivot table below showing the number each student got of each score, student 8 only got 5 instances of 1s.

Testing: Our null hypothesis (H_0) is that the variation of the score given to **student 8** by **volunteer 6** is due to chance. Our alternative hypothesis (H_a) is that the anomaly of the score is not due to chance and should be further investigated. The data we will be testing is the average rep score for student 8. We will assume a p-value of 0.05.

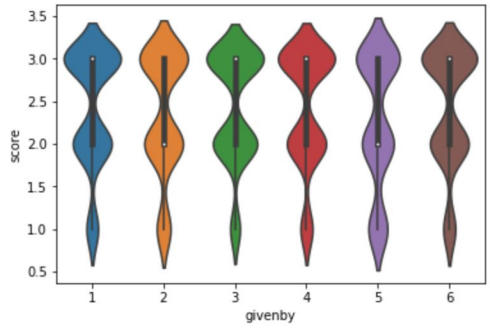
In python, we constructed violin plots to show the distribution of scores per student/per volunteer. Violin plots give the shapes of the distributions.

```
sns.violinplot(rep['SID'], rep['score'])
<matplotlib.axes._subplots.AxesSubplot at 0x7ef
```



We can see that most of the score distributions are similar; most get mostly 3s and 2s, although some students have more 1s like student 6. We can see that student 8 mostly has 3s with fewer 2s and barely any 1s.

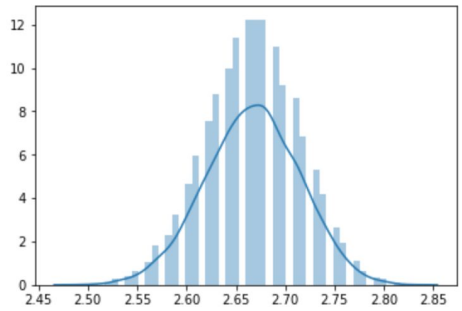
```
sns.violinplot(rep['givenby'], rep['score'])
<matplotlib.axes._subplots.AxesSubplot at 0x7efd6f
```



From this violin plot, we can see that all of the volunteers mostly give 3s and 2s with fewer 1s, which is consistent with the violin plots of the student scores. We can see that Volunteer 5 tends to give more 1s than the other volunteers, but Volunteer 6's scores are pretty consistent with the others.

We then bootstrapped the scores of student 8, taking out the score from volunteer 6, and took 10,000 averages of 100 scores. The distribution of the averages is shown in the plot below.

```
score86 = byBoth[8][6]
eight = rep[rep['SID'] == 8]['score']
randomScores = []

for x in np.arange(0, 10000):
    score = np.random.choice(eight, size=100, replace=True)
    randomScores.append(np.mean(score))

sns.distplot(randomScores)
```



From there, we constructed a 95% confidence interval and calculated the p-value of our sample.

```
rscores = pd.DataFrame(data = {'score': randomScores})
```

```
bottom5 = np.percentile(randomScores, .025)
top5 = np.percentile(randomScores, .975)

pval = len(rscores[rscores['score'] <= np.mean(byBoth[8])])/10000

print("middle 95% of mean scores from bootstrapping: (", bottom5, ",", top5, ")")
print("mean of student 8 with score from volunteer 6:", np.mean(byBoth[8]))
print("p-value:", pval)
```

```
middle 95% of mean scores from bootstrapping: ( 2.5177771002924754 , 2.5541610741398753 )
mean of student 8 with score from volunteer 6: 2.504629335
p-value: 0.0066
```

The predicted averages of student 8 from bootstrapping had a 95% confidence interval of about (2.5178, 2.5542) which does not include the mean of student 8, which is 2.504529335. The p-value we got was 0.0066 < 0.05, so we can reject the null hypothesis and conclude that the score may be an anomaly not due to chance, and pairing of student 8 and volunteer 6 should be further investigated.

Alternative Query 5:

Question: How to forecast facility capacity demand as program enrollment increases?
Data extracted: Select cumulative attendance for all program participants and total capacity across all courts in athletic facilities

SQL Query:

```
SELECT A.Date, COUNT(A.SID)
FROM Attendance AS A
Where substr(A.date, length(A.date)) = 30 OR substr(A.date, length(A.date)) = 28
Group BY A.Date;
```

Methodology: Time series analysis

Details and Justification:

SquashDrive should plan for an increasing program enrollment, and corresponding needs for available sports facility spots relative to total number of student participants. Utilize time series analysis through Autoregressive Integrated Moving Average (ARIMA) to find enrollment trend patterns (MoM), and extrapolate demand predictions.

The tool we used to develop the model and analysis was Rstudio. Below on the left we can see the data outputted from the SQL query, while on the right it is the detailed RScript for creating the model. Notice since for our database, attendance is measured on a daily basis, and predicting one day ahead does not really serve any meaningful results for the organization, thus we have decided to only choose the attendance at the end of each month to make a model for predicting on a monthly basis. Additionally, our database doesn't really keep track of the total number of student enrollment OVER time, thus we used attendance as a close estimator for the total number of students enrolled at that time.

Data Extracted (exported to Excel):

Date	Attendance
1/30/2017	49
2/28/2017	50
3/30/2017	55
4/30/2017	51
5/30/2017	49
6/30/2017	52
7/30/2017	52
8/30/2017	55
9/30/2017	51
10/30/2017	58
11/30/2017	56
12/30/2017	58
1/30/2018	56
2/28/2018	58
3/30/2018	55
4/30/2018	58

R Code :

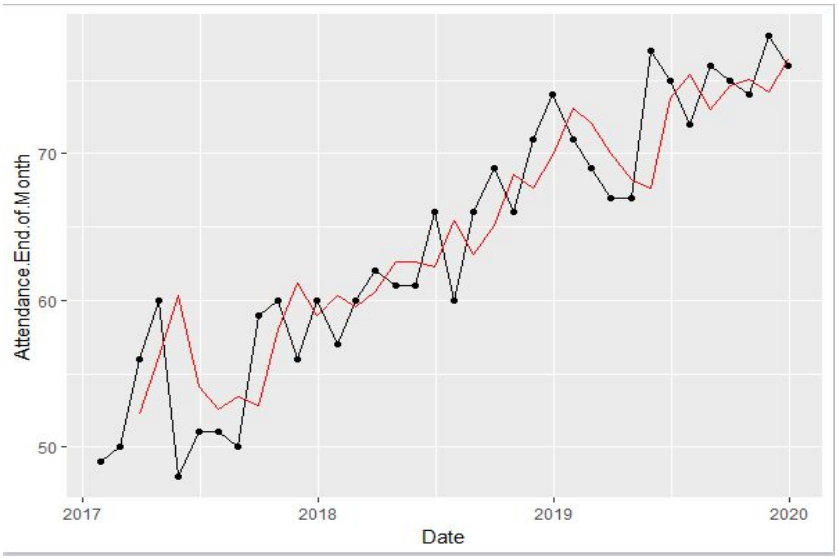
```
library(dplyr)
library(ggplot2)
library(lubridate)

Attendance <- read.csv("TimeSeries.csv")
Attendance <- Attendance %>% mutate(Date = mdy(Date))

AttendanceTrain <- Attendance %>% filter(year(Date) < 2020)
AttendanceTest <- Attendance %>% filter(year(Date) == 2020)

ggplot(Attendance, aes(x=Date, y=Attendance.End.of.Month)) +
  geom_line() +
  geom_point()

AttendanceTrain <- AttendanceTrain %>%
  mutate(AttendanceLastMonth=c(NA, head(Attendance.End.of.Month, -1))) %>%
  mutate(AttendanceTwoMonthAgo = c(NA, NA, head(Attendance.End.of.Month, -2)))

TimeSeries <- lm(Attendance.End.of.Month~AttendanceLastMonth + AttendanceTwoMonthAgo, data=AttendanceTrain)
summary(TimeSeries)

ggplot(AttendanceTrain, aes(x=Date, y=Attendance.End.of.Month)) +
  geom_line() +
  geom_point() +
  geom_line(aes(y=predict(TimeSeries, newdata=AttendanceTrain)), col="red")
```

Now the model that we have created is ARMA, Auto-Regressive Moving Average. The main idea of this model, in terms of our project, is that the predicted attendance for next month is a **linear combination** between the average attendance of past months and the attendance for the current month. Thus, we fitted a linear regression model(adding on another variable of attendance two month back to see if the model finds it significant too) where the constant term is treated as the expected mean.

Below shows the result of our model, we can see that from the data we created it, the overall significance of the model helps us conclude that there is a relationship between our variables and the dependent variable. We can see that that the AttendanceLastMonth variable was the only one that was very significant, which makes sense in our case as we just choose random numbers for the data and we incremented the random range at each step.

```
Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)           7.8933     5.5639   1.419   0.1660
AttendanceLastMonth   0.6135     0.1721   3.565   0.0012 **
AttendanceTwoMonthAgo 0.2788     0.1720   1.621   0.1152
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.186 on 31 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.7748,    Adjusted R-squared:  0.7603
F-statistic: 53.34 on 2 and 31 DF,  p-value: 9.205e-11
```

And below here we can see the actual data vs predicted attendance from a month by month basis.



In conclusion, we believe that having a model to predict enrollment increase can be fruitful for the organization. Not only to prepare when to expand on court capacities, but also for other internal planning purposes for events, activities, and even classroom management. With that said, we want to emphasize on that this model is still young as it is only fed with the preliminary data(random numbers), but through implementing it with real data and assessing the model's

performance, we believe that this will be another great asset for the organization on top of the whole database system we have designed.

Normalization Analysis:

Relation #1:
Partner School (Name, Address, Person of contact, List of students in the program)

Functional Dependencies:
{Name, Neighborhood} → {Person of contact, List of students in the program}

The relation has multi-valued attributes, therefore, it is not in 1NF (UNF).

To normalize to 1NF, all of our attributes have to be single-valued. This is done by creating two relations:

Partner School (Name, Neighborhood, Person of Contact)
Student Info (Name, Neighborhood, Student Name)

The relations are already in both 2NF and 3NF after normalizing to 1NF. They are in 2NF because every non-prime attribute is fully dependent on the key. They are in 3NF because no non-prime attribute is transitively dependent on the key. However, since person of contact, a non-prime attribute, determines the name of the partner school, a prime-attribute, the relation is not in BCNF.

For example, for the relation "Partner School", let
T1[Berkeley Elementary, Elmwood, Jane]
T2[Berkeley Elementary, Downtown Berkeley, Chris]
T3[Berkeley Elementary, Emeryville, Jane] be three tuples.
The key, the name of the school and the neighborhood, determines person of contact, while person of contact determines the name of the school, which violates BCNF.

To normalize to BCNF:

Partner School (Neighborhood, Person of Contact)
School Contact (Person of Contact, Name)
Student Info (Name, Neighborhood, Student Name)

The new relations are in BCNF as well because for all functional dependencies X → Y, X is a superkey.

Relation #2:
Equipment (EQID[8], Name, Cost, Date purchased, Equipment type, Years, Condition)

Functional Dependencies:
{EQID} → {Name, Cost, Date purchased, Equipment type, Number of years in use, Condition}
{Date purchased} → {Number of years in use}

{Name} → {Equipment type, Cost}

The relation is currently in 2NF since every non-prime attribute is fully dependent on the key. However, it's not in 3NF because there exists non-prime pairs (date purchased and number of years in use) and (name and equipment type, cost) such that date purchased determines number of years in use and name determines both equipment type and cost of equipment.

To normalize to 3NF:
R1(EQID, Name, Date purchased, Condition)
R2(Date purchased, Number of years in use)
R3(Name, Equipment type, Cost)

The relations are now both in 3NF and BCNF. They are in BCNF as well because for all functional dependencies X → Y, X is a superkey.

Relation #3:
Donation (Donor[1e], Date, Type, Amount)

Functional Dependencies:
{Donor, Date} → {Type, Amount}

The relation is already in BCNF, because for all functional dependencies X → Y, X is a superkey.

Relation #4:
SponsorFundsEvent (SPID, EventID[4], Amount, Date of Transaction)

Functional Dependencies:
{SPID, EventID} → {Amount, Date of Transaction}
{Date of Transaction} → {EventID}

The relation is in 2NF because every non-prime attribute is fully dependent on the key. EventID alone does not determine date of transaction because multiple transactions can be made for the same event on different dates by different SPIDs. It is also in 3NF because no non-prime attribute is transitively dependent on the key. However, it is not in BCNF because a non-prime attribute, date of transaction, determines a prime attribute, eventID. This is because fundings can only be made to the nearest event.

To normalize to BCNF:
R1(Date of Transaction, EventID)
R2(SPID, Date of Transaction, Amount)

The relations are now in BCNF because for all functional dependencies X → Y, X is a superkey.

Relation #5:

Grade (<u>Course Number</u>, <u>Semester</u>, <u>SID</u>[1a], Teacher Name, Grade)

Functional Dependencies:
{SID, Semester, Course Number} → {Grade}
{Semester, Course Number} → {Teacher Name}

The relation is in 1NF because all the attributes are single-valued. However, it is not in 2NF because a subset of the key, semester and course number, determines a non-prime attribute, teacher name.

To normalize to 2NF:

Grade1 (<u>SID, Semester, Course Number,</u> Grade)
Grade2 (<u>Semester, Course Number,</u> Teacher Name)

The relations are now in 3NF and BCNF as well. They are in 3NF because no non-prime attributes are transitively dependent on the key. They are also in BCNF because for all functional dependencies X → Y, X is a superkey.