NIA HOUSE.
Montessori toddler and preschool Program

Database Project
Berkeley Engineering

Team 2
Hoi Ping Chan, Tsoiling Chung, Erin Davis, Carrina Lai, Qin Liang, Yasaman Nazemitarzizi, Ferina Putri, Sio Fong Wong, Lu Zhao
1. Toddler
2. Pre-School

1. 10 employees
2. 50 students
**REVIEW.  Advantages of Our Database**

**Old Database Structure**
- Different files on students
- Student performance
- Data on registration & billing info

**Problems**
- Duplication of data
- Wasted storage space
- Inconsistency

**Our database**
- Control of redundancy
- Integrates different files into one
- Eliminate above problems
- Retrieve & update related data
II

student evaluation

Program

(1,n)

(0,n)

receives

enrolls in

requests

toddler evaluation

pse: creative expression

pse: language

pse: math

pse: motor skills

pse: sensorial

pse: premath/prelanguage

pse: practical life

pse: physical sciences

pse: nutrition

student evaluation

d,t
1. Student (SID, Lname, Fname, MI, Birthdate, Sex, GID, Address, phone#, family history(att))
   1a. Current_student (SID, ..., Health_history (attachment),
       Physician_report(att), Enrollment_date)
   1b. Prospective.student (PSID\(^1\), Waitlist#, ApplyDate, InitialWaitNum,
       StudentName, StudentBdate, GuardianName, Address, FamilySize,
       GrossYearlyIncome, HowYou Heard About Us)
   1c. Alumnus (Lname, Health_history (attachment),
       Physician_report(att), Graduation_date)
2. Staff (SSN, Lname, Fname, MI, Birthdate, Phone#, Sex, Street, ..., Email, Salary,
       Supervisor_SSN\(^2\))
   2a. Teacher (SSN, ..., Program, Shift_start, Shift_end)
   2b. Volunteer (SSN, ..., Job)
   2c. Admin (SSN, ..., Shift_start, Shift_end)
   2d. Fundraising_officer (SSN, Shift_start, Shift_end, ..., Project)
3. Donor (DID, SSN, Lname, ..., Donations)
4. Donation (DonID, Time, Date)
   4a. FundAmt (DonID\(^4\), Time, Date, ..., Amount, Payment_method)
   4b. Item (DonID\(^4\), Time, Date, ..., Supply_type\(^6\))
5. Guardian (GID, Lname, Fname, Address, Phone#, Income)
6. Supply (Stype, Cost, Quantity, Donated, Maintenance, Classroom, Equipment)
7. Scholarship (Scholar_name, Amount, Start_date, End_date, Source)
8. Fundraising_project (Fund_name, Date, Amount_raised, Start_date, End_date, Strategy)
9. Participate (DID, Fund.name, Amount_given)
10. Manage (SSN, Fund.name)
11. Timeslot (TimeSlotID, Start_hour, Date)
12. Financial Transaction (TID, Time, Date, Amount, Account#)
   12a. Withdraw (TID)
   12b. Deposit (TID, GID, DonID)
13. StudentEval (EvalID, SID, Date, Evalattachment)
14. EvalJunction (TeacherSSN, EvalID)
15. Receive (SID, Scholar_name)
16. Fund (TID^{12}, Scholar_name^7)
17. Check_In (TimeslotID, SID^1)
18. Check_Out (TimeSlotID, SID^1)
19. Facilities (Location, Name, Landlord, Monthly_payment, Insurance)
20. Require (TID^{12}, Stype^6)
21. Pay_salary (TID^{12}, SSN^2)
22. Pay_to (TID^{12}, Facility_name^19)
23. Schedule (StaffSSN^2, TimeSlotID^{11})-junction
24. Contribute(DID_3, DonID_4)
25. Program(ProgID, description, EnrolledStudent^1)
26. Teach(ProgID^{25}, Teacher^{2a})
27. Takes_place_during(ProgID, TimeSlotID)
28. Relate(GID^5, SID^1)
Team 2

Review
Norm
Query

Nia House
NORMALIZATION.

1NF

PreschoolStudentEval(SID, EvalID, EvalDate, TeacherSSN, Notes)

SID, EvalID → EvalDate, TeacherSSN, Notes
SID → TeacherSSN

Class(TeacherSSN, SID)
PreschoolStudentEval(SID, EvalID, EvalDate, Notes)

2NF

Student(SID, GuardianID, Lname, Fname, MI, Bdate, Sex, Address, Phone#, WaitDateAdded, DateEnrolled)

SID → GuardianID, Lname, Fname, MI, Bdate, Sex, Address, Phone#, WaitDateAdded, DateEnrolled
GuardianID → Address, Phone#

Student(SID, GID, Lname, Fname, MI, Bdate, Sex, Address, WaitDateAdded, DateEnrolled)
GuardianContactInfo(GID, Phone#, Address)

Donor(DID, SSN, Lname, Fname, CompanyName)

DID → SSN, Lname, Fname, Cname
SSN → Lname, Fname

Donor(DID, SSN, CompanyName)
DonorName(SSN, Lname, Fname)
BCNF

Guardian(GID, Lname, Fname, Phone#, Address, Income)
GID → Lname, Fname, Phone#, Address, Income

Staff(SSN, Lname, Fname, MI, Bdate, Phone#, Sex, Email, Salary, SupervisorSSN)
SSN → Lname, Fname, MI, Bdate, Phone#, Sex, Email, Salary, SupervisorSSN
Average time on waitlist

Function
- Calculate the waiting time for each student on the waitlist

Purpose
- Clients can estimate how long the student may have to wait before enrolling

<table>
<thead>
<tr>
<th>PSID</th>
<th>StudentName</th>
<th>DaysTillAccepted</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>David Garrett</td>
<td>12.70</td>
</tr>
<tr>
<td>18</td>
<td>Henry Wessle</td>
<td>25.41</td>
</tr>
<tr>
<td>20</td>
<td>Ophelia Westwooo</td>
<td>38.11</td>
</tr>
<tr>
<td>21</td>
<td>Ermine Porter</td>
<td>50.81</td>
</tr>
<tr>
<td>22</td>
<td>Nancy Carter</td>
<td>63.52</td>
</tr>
<tr>
<td>23</td>
<td>Jake Sullivan</td>
<td>76.22</td>
</tr>
<tr>
<td>24</td>
<td>Helen Cysawyre</td>
<td>88.92</td>
</tr>
<tr>
<td>1</td>
<td>Eve Holland</td>
<td>101.63</td>
</tr>
<tr>
<td>2</td>
<td>Amy Suki</td>
<td>114.33</td>
</tr>
<tr>
<td>3</td>
<td>Harry Truesdale</td>
<td>127.03</td>
</tr>
<tr>
<td>4</td>
<td>Vivian Long</td>
<td>139.74</td>
</tr>
</tbody>
</table>
Average time on waitlist.

SELECT PSID, StudentName, WaitNum*davg("((Date()-ApplyDate)/ (InitialWaitNum-WaitNum))", "Prospective Student") AS DaysTillAccepted
FROM [Prospective Student];
Trends in # of waitlist.

**Function**
- To forecast the **expected trends** in coming seasons to determine the class size using a 4\(^{th}\) degree polynomial regression
- To find out which month in the season has the **most number** of prospective students

**Purpose**
- Nia House can use the data to forecast how many students may wish to enroll in and accordingly plan for additional teachers, equipment, and facilities
SELECT Month(ApplyDate) AS [month], Count([Prospective Student].PSID) AS TotalStu, Year(ApplyDate) AS [Year] FROM [Prospective Student] GROUP BY Month(ApplyDate), Year(ApplyDate);


Trends in # of waitlist.
# Students' Performance

## Creative Expression

<table>
<thead>
<tr>
<th>Activity</th>
<th>Level of Participation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participates in art works</td>
<td>1 2 3 4</td>
</tr>
<tr>
<td>Enjoys product-oriented art activities</td>
<td>1 2 3 4</td>
</tr>
<tr>
<td>Enjoys process-oriented art activities</td>
<td>1 2 3 4</td>
</tr>
<tr>
<td>Shares on share day</td>
<td>1 2 3 4</td>
</tr>
<tr>
<td>Expresses ideas and thoughts to adults and children</td>
<td>1 2 3 4</td>
</tr>
<tr>
<td>Expresses ideas and thoughts to a large group</td>
<td>1 2 3 4</td>
</tr>
<tr>
<td>Participates in movement and music activities</td>
<td>1 2 3 4</td>
</tr>
<tr>
<td>Plays music/rhythms with instruments</td>
<td>1 2 3 4</td>
</tr>
<tr>
<td>Dramatic play with friends</td>
<td>1 2 3 4</td>
</tr>
</tbody>
</table>

## Motor Skills

- Runs
- Skips
- Gallops
- Hops with 1 foot
- Hops with 2 feet
- Climbs the play structures
- Throws a ball
- Catches a ball
- Kicks a ball
- Swings a bat (crosses midline)
- Pedals a bike
- Balances on a beam

## Practical Life

- Follows classroom rules
- Follows playground rules
- Works, shares, plays cooperatively
- Works independently
- Works with concentration
- Completes the work cycle
- Works with spatial order
- Respects others
- Works cooperatively with others
- Follows oral directions
Function
Calculate the average of student’s evaluation in each skill area over time.

Purpose
- Provide a better visualization of student’s performance, so teachers can notice the trend in each student’s performance
- Find high performing and low performing students
- Find low performing topics
Students' Performance.

Code to find the average grade for each skill:

```
SELECT SID, EvalID, 
(Nz([1],0)+Nz([2],0)+Nz([3],0)+Nz([4],0)+Nz([5],0)+Nz([6],0)+Nz([7],0)+Nz([8],0)+Nz([9],0)) AS total, 
IIf(IsNull([1]),0,1)+IIf(IsNull([2]),0,1)+IIf(IsNull([3]),0,1)+IIf(IsNull([4]),0,1)+IIf(IsNull ([5]),0,1)+IIf(IsNull([6]),0,1)+IIf(IsNull([7]),0,1)+IIf(IsNull([8]),0,1)+IIf(IsNull([9]),0,1 ) AS [count], [total]/[count] AS Average 
FROM [PreStudentEval Creative Expression];
```

Above is the example for Creative Expression, we use this code for other skills as well.
QUERY Financials
3 Annual Cash Flow.

**Function**
- Graph monthly total cash inflow and cash outflow
- Graph the breakdown of the inflow from tuition and donations

**Purpose**
- Provides the cash inflows and outflows for a period (monthly here)
- Breakdowns of the contributions from tuition payments and donations
1. Monthly Total/Max Cash Inflow
   SELECT TYear, TMonth, sum([ft1].Amount) as Total_In, Max([ft1].Amount) as max_inflow
   FROM [Financial Transaction] as [ft1], Deposit as d1
   WHERE [ft1].TID = d1.TID
   GROUP BY TYear, TMonth;

2. Monthly Total/Max Cash Outflow
   SELECT TYear, TMonth, sum([ft2].Amount) as Total_out, Max([ft2].Amount) as max_outflow
   FROM [Financial Transaction] as [ft2], withdraw as w1
   WHERE [ft2].TID = w1.TID
   GROUP BY TYear, TMonth;

3. Monthly Breakdown - Guardian
   SELECT [ft1].Tyear, [ft1].Tmonth, Sum([ft1].Amount) AS SumOfGID
   FROM Deposit AS d1, [Financial Transaction] AS [ft1]
   WHERE (((d1.GID) Is Not Null) AND ((d1.TID)=[ft1].[TID]))
   GROUP BY [ft1].Tyear, [ft1].Tmonth;

4. Monthly Breakdown - Donner
   SELECT [ft1].Tyear, [ft1].Tmonth, Sum([ft1].Amount) AS SumOfDonID
   FROM Deposit AS d1, [Financial Transaction] AS [ft1]
   WHERE (((d1.DonID) Is Not Null) AND ((d1.TID)=[ft1].[TID]))
   GROUP BY [ft1].Tyear, [ft1].Tmonth;
Objective

Balance purchasing and storage costs using EOQ

Determine the order quantity that minimized the total annual cost using EOQ

Define the inventory level that will trigger an order using (ROP)

Function

- Review current supply level
- Know optimal order quantities and the next order date

Purpose

- Minimize purchasing and inventory costs
EOQ (Q*) = $\sqrt{\frac{2 \times \text{Ordering cost} \times \text{Daily Demand}}{\text{Interest Rate} \times \text{Unit Cost}}}$

Reorder Point (ROP) = Daily Demand * Lead Time

Assumption
- Daily Demand is known and constant
- Delivery lead time is known and constant
- Safety stock needed when lead time is not zero
- Ordering cost, unit cost, and interest rate are known and constant
- 365 school days
- No quantity discounts
- Purchase date updated whenever an order placed
4 Supply Management.

SELECT SupplyID, SupplyName, IIF(CurrentInventory=0,'Yes','No') AS Stockout, round(sqr(2*[OrderingCost]*[DailyDemand]/(InterestRate*UnitCost))) AS OptimalOrderQ, IIF(LeadTime>0, round(LeadTime*[DailyDemand]/(InterestRate*UnitCost)),0) AS ReorderPoint, round(((sqr(2*[OrderingCost]*[DailyDemand]/([InterestRate]*[UnitCost])))/[DailyDemand])*365) AS OrderCycleInDays, IIF([CurrentInventory]=0, Date(), ([PurchaseDate]+round(((sqr(2*[OrderingCost]*[DailyDemand]/([InterestRate]*[UnitCost])))/[DailyDemand])*365))-LeadTime) AS NextOrderDate
FROM Supply
ORDER BY IIF([CurrentInventory]=0, Date(), ([PurchaseDate]+round(((sqr(2*[OrderingCost]*[DailyDemand]/([InterestRate]*[UnitCost])))/[DailyDemand])*365)));
Fundraising Analysis
Function

- Perform a linear regression on date vs. amount raised on each fundraising event
- Return the correlation coefficient, slope, and intercept of the best fit line
- Output the response rate of donors to each fundraising event.

Purpose

- Prove a variety of information on Nia House fundraisers
- Find out which fundraising event raises the highest amount of money
- Find out the best method to reach potential donors
1. **Fundraising Donors’ Information**
   ```sql
   SELECT d.Lname, d.Fname, d.Cname, dsum("amountgiven", "participate", "participate.DID ="&[d.DID]) AS TotalGiven, dcount("amountgiven", "participate", "participate.DID ="&[d.DID]) AS TimesDonated, davg("amountgiven", "participate", "participate.DID ="&[d.DID]) AS AverageDonation FROM donor AS d;
   ```

2. **Fundraising regression**
   ```sql
   SELECT fp.Strategy, (Count(Date)*Sum(Date* [Amount Raised])-Sum(Date)*Sum([Amount Raised]))/(Sqr((Count(Date)* Sum(Date^2)-(Sum(Date)^2))*(Count([Amount Raised])*Sum([Amount Raised]^2)-(Sum([Amount Raised])^2)))) AS R, R*stdev([Amount Raised])/stdev(Date) AS B, avg([Amount Raised])-B*avg(Date) AS A FROM [Fundraising Project] AS fp GROUP BY fp.Strategy;
   ```

3. **Response Rate**
   ```sql
   SELECT f.[Fund Name], f.FID, dcount("DID", "participate", "participate.FID ="&[f.FID])/dcount("DID", "donor") AS ResponseRate FROM [Fundraising Project] AS f;
   ```
Thank you!

Individual Donations
Donations vs. Time

Function
- To graph the monthly item donations and the monthly money donations respectively
- To perform Chi-Square Goodness of Fit Test and see whether numbers of donations are consistent over a year
- To perform ANOVA and see whether different months have an effect on the amount of money received from every donation

Purpose
- To analyze the random donations that do not come from fundraisers
- To find out whether time affects the amount of donations and also the type of donations—money or item
1. **Expected Value**
   
   ```sql
   SELECT ((max(N.[No# of donations])+ min(N.[No# of donations]))/2) AS [Expected Value]
   FROM [Q5 - no# of all donations] AS N;
   ```

2. **Test Statistics**
   
   ```sql
   SELECT Sum(((N.[No# of donations]-E.[Expected Value])^2)/E.[Expected Value]) AS [Test Statistics], Count(N.[No# of donations])-1 AS df
   FROM [Q5 - no# of all donations] AS N, [Q5 - chi - Exp value] AS E;
   ```
1. **Total Average**
   
   ```sql
   SELECT count(*) AS [no# of $ donations], avg(M.Amount) AS [Average amount]
   FROM [Q5 - money - donations list] AS M;
   ```

2. **Sum of square, mean square, df and F-ratio result**

   ```sql
   SELECT M.TMonth, Var(M.Amount)*((Count(M.TMonth)-1) AS SumDiff
   FROM [Q5 - money - donations list] AS M GROUP BY M.TMonth;
   
   SELECT Sum(S.[No# of $ donations]*S.[Average amount]-T.[Average amount])*S.[Average amount]-T.[Average amount]) AS [SS treatment], Count(S.Month-1) AS [df treatment], [SS treatment]/11 AS [MS treatment], Sum(E.SumDiff) AS [SS error], (Sum(S.[No# of $ donations])-12) AS [df error], [SS error]/(Sum(S.[No# of $ donations])-12) AS [MS error], [MS treatment]/[MS error] AS [F Ratio]
   WHERE (((S.Month)=E.[TMonth]));
   ```

---

**ANOVA**

<table>
<thead>
<tr>
<th>Yearly summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>no# of $ donations</td>
</tr>
<tr>
<td>24</td>
</tr>
</tbody>
</table>

**ANOVA result**

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SS treatment</td>
<td>$33,733.33</td>
<td>df treatment</td>
<td>12</td>
<td>SS error</td>
<td>61800</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MS treatment</td>
<td>3066.666663636363</td>
<td>df error</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MS error</td>
<td>5150</td>
<td>F Ratio</td>
<td>0.595469255075022</td>
</tr>
</tbody>
</table>
### Purpose
- To analyze the random donations that do not come from fundraisers
- To find out whether time affects the amount of donations and also the type of donations—money or item

### Function
- To graph the monthly item donations and the monthly money donations respectively
- To perform Chi-Square Goodness of Fit Test and see whether numbers of donations are consistent over a year
- To perform ANOVA and see whether different months have an effect on the amount of money received from every donation

### 1. Item donations list
```sql
SELECT D.*, I.SupplyName
FROM Donation AS D, Item AS I
WHERE D.DonationID = I.DonationID AND D.DYear=2012;
```

### 2. Money donations list
```sql
SELECT *
FROM Deposit AS D, [Financial Transaction] AS F
WHERE D.DonationID IS NOT NULL AND D.TID=F.TID AND F.TYear=2012;
```

### 3. Summary for each month for Money donations
```sql
SELECT F.TMonth AS [Month in 2012], count(F.TMonth) AS [No# of $ donation],
sum(F.Amount) AS [Total amount], avg(F.Amount) AS [Average amount]
FROM Deposit AS D, [Financial Transaction] AS F
WHERE D.DonationID IS NOT NULL AND D.TID=F.TID AND F.TYear=2012
GROUP BY F.TMonth;
```

### 4. No# of donations in each month
```sql
SELECT D.DMonth AS [Month in 2012], count(D.DMonth) AS [No# of donations]
FROM Donation AS D WHERE D.DYear=2012
GROUP BY D.DMonth;
```
Thank You!

Q & A