Agenda

I. Business Overview
II. EER Diagram
III. Relational Schema
IV. Queries
V. Normalization Analysis
A place to practice yoga throughout the day, every day with some of the Bay Area's finest and most experienced teachers.
**Power**
Sun salutations and **flowing standing and twisting sequences.** Includes Savasana

**Vinyasa**
Flow based practice and unique sequencing. Poses may be held for longer periods.

**Hot**
Taught in a **heated room** from 95-100 degrees and gives students a unique experience

**Pricing**
- New Student Intro: $30
- Drop-In: $15
- 5-Class Series: $65
- 10 Class Series: $120
- Monthly Unlimited: $150

*“Yoga is a practice for body, mind and spirit. The mix will be different from class to class, based on how you’re feeling.”*
Project Progression

DP 1
- Simplified EER design with 30 entities and relationships
- Familiarization with Groove Yoga
- Created project timeline

DP 2
- Refined EER design by adding new attributes and relationships while deleting unnecessary ones
- Conceptualized five queries
- Relational design schema
- Created Access database

DP 3
- SQL implementation of queries
- Access implementation of queries
- Relations defined in database
- Major query revision

Final Project
- Queries refined and finalized in Access
- Finalized relationships in Access
- Normalization analysis
Key Relations

Salary (InPID\textsuperscript{15}, SalaryPerClass, OwPID\textsuperscript{19}-Owner)

Class (CID, RID\textsuperscript{7}, Type, FrPID\textsuperscript{14}, TimeID\textsuperscript{8}, Month, Date, Year)

FacilityCosts (BillId, FacID\textsuperscript{32}, Month, Year, TypeOfCost, PaidToCompany, AmountPaid)

Client (CIPID\textsuperscript{10}, Credit CARD Number, SkillLv)

YogaPackage (PackageID, MonthSold, DateSold, YearSold, DateExpires, 
PackageName, ClassesAvailable, PID\textsuperscript{10}-Owns, Cost)

Optimize pricing for introductory student special

Determine effectiveness of introductory student special
Key Relations

**Person** (PID, Fname, MI, Lname, Address, ZIP, City, State, Phone_Number, Join_Date, Emergency_Contact_Name, Emergency_Contact_Number, DOB, Email, Gender, Age, Student_Status)

**PackageFinancialTransaction** (TID, PackageID, PID-Purchases, FrPID-Sells, Month, Date, Year, PurchasePrice)

**ProductFinancialTransaction** (TID, PID-Purchases, PID-Sells, IID, Month, Date, Year, PurchasePrice)

**InstructorEvaluation** (EID, Score, Date, PID-Client, PID-Instructor)

**AttendsClass** (CID, CID, PackageID)
Query 1

How do we prevent people from sharing their unlimited monthly passes?

A.I. Facial Recognition

Client enters Phone # Gets Picture Scanned Confirmed or Denied
How It Works

[SQL/VBA] Retrieves stored customer photo taken at registration based on inputted phone number

[SIFT] Convert stored photo and the new photo the client scans in to descriptor and detector matrices

[vl_ubcmatch] Calculate the number of matches between the two photographs and “scores” on how similar they are based on squared Euclidian distances
The Technical Details

1. **SQL/VBA**

   ![FacialRecognitionForm](image)

   **Download Image**

   **Phone Number**: 1767556465

2. ```vba
Private Sub Command0_Click()
    Dim rs As DAO.Recordset
    Dim strSQL As String
    Dim dbs As Database
    Dim fld As Field
    Dim strIntroName, strPromoName As String

    Set dbs = CurrentDb
    strSQL = "SELECT Person.Attachment.FileName, Person.Attachment.FileData, Person.* FROM Person WHERE " & 
             "((Person.Phone_Number)=" & Forms!FacialRecognitionForm!PhoneNum & ")"

    Set rs = dbs.OpenRecordset(strSQL)
    SaveAttachments rs
End Sub"```
Function SaveAttachments(ByVal rstCurrent As DAO.Recordset)
    Const CALLER = "SaveAttachments"
    On Error GoTo SaveAttachments_ErrorHandler
    Dim rstChild As DAO.Recordset2
    Dim fldAttach As DAO.Field2
    Dim strFilePath, strFieldName, strOutputDir, m_strFieldFileName As String
    Dim m_strFieldFileData As String
    
    strFieldName = "Attachment"
    strOutputDir = SelectSaveAs("Desktop")
    m_strFieldFileName = "LogFile"
    m_strFieldFileData = CStr(rstCurrent.Fields(0))
    m_strFieldFileData = "FileData"
    
    If Right(strOutputDir, 1) <> "\" Then strOutputDir = strOutputDir & "\\"
    Set rstChild = rstCurrent.Fields(strFieldName).Value
    strFilePath = strOutputDir & m_strFieldFileName
    If Dir(strFilePath) <> "" Then
        VBA.SetAttr strFilePath, vNormal
        VBA.Kill strFilePath
    End If
    Set fldAttach = rstChild.Fields(m_strFieldFileData)
    fldAttach.SaveToFile strFilePath
    rstChild.Close ' cleanup
    MsgBox "File Saved", vbOKOnly
    Exit Function
SaveAttachments_ErrorHandler:
    Debug.Print "Error ": Err.Number & " in ": CALLER & ": " & Err.Description
    MsgBox Err.Description, vbMsgBoxStyle.vbCritical, "Error ": Err.Number & " in ": CALLER
    Debug.Assert False
    Resume Next
End Function

Overview ➤ EER ➤ Schema ➤ Queries ➤ Normalization
The Technical Details

[SIFT]

1. **Scale-space extrema detection**: searches over image locations using a difference-of-Gaussian function to identify potential interest points.

2. **Keypoint localization**: At each interest point, a model is fit to determine location and scale based on stability.

3. **Orientation assignment**: Orientations are assigned to each keypoint location based on image gradient directions.

4. **Keypoint descriptor**: the image gradients are measured within the region and transformed into matrix representation.

```matlab
if margin == 0
    im1 = imread('p1.jpg');
    im2 = imread('p2.jpg');
end

% make single
im1 = im2single(im1);
im2 = im2single(im2);

% make grayscale
if size(im1,3) > 1, im1g = rgb2gray(im1); else im1g = im1; end
if size(im2,3) > 1, im2g = rgb2gray(im2); else im2g = im2; end

[f1,d1] = vl_sift(im1g);
[f2,d2] = vl_sift(im2g);

thresh = 2.0;
(matches, scores) = vl_ubcmatch(d1, d2, thresh);

indices1 = matches(1,:);
f1match = f1(:,indices1);
d1match = d1(:,indices1);

indices2 = matches(2,:);
f2match = f2(:,indices2);
d2match = d2(:,indices2);
```
The Technical Details

**VL_UBCMatch**

```plaintext
[f1,d1] = vl_sift(img1);
[f2,d2] = vl_sift(img2);
thresh = 2.0;
[matches, scores] = vl_ubcmatch(d1, d2, thresh);
```

For descriptor d1 inputted, the closest descriptor in d2 is found. The index of which descriptors in d1 match which descriptors in d2 is stored into “matches”.

The match needs to be significantly better than any other match with Significant defined in threshold.

The squared Euclidean distance between the matches are stored in “scores”.

Overview   EER   Schema   Queries   Normalization
What client characteristics increase likelihood of conversion?

Rate of Conversion

- Analyze Client Attributes
- Determine Conversion Rates
- Focus Marketing Efforts
How It Works

[SQL/VBA]  [RStudio]

Retrieves all reviews clients have made in the past towards instructors based on categories

Retrieves recommended clients based on factors of interest to examine where to focus marketing efforts to drive conversion rates up
The Technical Details

[SQL]

```
```

Returns the client PID and related factors that might influence their decision to convert or buy a long-term package.
The Technical Details

[SQL/VBA]

Checks if the client has bought a package after the introductory student special

```vbnet
Public Function FieldExists(newPID As Integer) As Boolean

Dim rs As DAO.Recordset
Dim strSQL As String
Dim dbs As Database
Dim fld As Field
Dim strIntroName, strPromoName As String

strIntroName = "Intro Package"
strPromoName = "Promotional Pass"

Set dbs = CurrentDb
FieldExists = False

strSQL = "SELECT yp1.[PID-Owns] & _
    "FROM YogaPackage AS yp1 INNER JOIN YogaPackage " & _
    "AS yp2 ON yp1.[PID-Owns] = yp2.[PID-Owns] " & _
    "WHERE ((yp1.PackageName) = & Chr$(39) & strIntroName & Chr$(39) & ") AND " & _
    "((yp2.PackageName) <> & Chr$(39) & strIntroName & Chr$(39) & ") AND " & _
    "((yp2.PackageName) <> & Chr$(39) & strPromoName & Chr$(39) & ") AND " & _
    "((yp1.[PID-Owns]) = & newPID & ");"

Set rs = dbs.OpenRecordset(strSQL)

If rs.EOF Then
    FieldExists = True
Else
    FieldExists = False
End If

End Function
```
## The Technical Details

### Access Output

<table>
<thead>
<tr>
<th>CIPID</th>
<th>Age_Group</th>
<th>StudentStat</th>
<th>ZIP</th>
<th>AvgScore</th>
<th>CountScore</th>
<th>CountClasse</th>
<th>Converted</th>
</tr>
</thead>
<tbody>
<tr>
<td>5056</td>
<td>25 - 34</td>
<td></td>
<td>94608</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>5057</td>
<td>25 - 34</td>
<td></td>
<td>94608</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5058</td>
<td>25 - 34</td>
<td></td>
<td>94608</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5059</td>
<td>35 - 44</td>
<td></td>
<td>94608</td>
<td>8.5</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5060</td>
<td>45 - 54</td>
<td></td>
<td>94608</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5061</td>
<td>35 - 44</td>
<td></td>
<td>94608</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5062</td>
<td>Over 65</td>
<td></td>
<td>94608</td>
<td>7.5</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5063</td>
<td>Under 18</td>
<td></td>
<td>94608</td>
<td>7.5</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5064</td>
<td>18 - 24</td>
<td></td>
<td>94608</td>
<td>7.5</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5065</td>
<td>18 - 24</td>
<td></td>
<td>94608</td>
<td>7.5</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5066</td>
<td>18 - 24</td>
<td></td>
<td>94608</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5067</td>
<td>35 - 44</td>
<td></td>
<td>94608</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5068</td>
<td>35 - 44</td>
<td></td>
<td>94608</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5069</td>
<td>45 - 54</td>
<td></td>
<td>94608</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5070</td>
<td>35 - 44</td>
<td></td>
<td>94608</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5071</td>
<td>45 - 54</td>
<td></td>
<td>94608</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5072</td>
<td>45 - 54</td>
<td></td>
<td>94608</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5073</td>
<td>35 - 44</td>
<td></td>
<td>94608</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5074</td>
<td>Over 65</td>
<td></td>
<td>94608</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5075</td>
<td>45 - 54</td>
<td></td>
<td>94608</td>
<td>8</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5076</td>
<td>Over 65</td>
<td></td>
<td>94608</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5077</td>
<td>Over 65</td>
<td></td>
<td>94608</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5078</td>
<td>Over 65</td>
<td></td>
<td>94608</td>
<td>8</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5082</td>
<td>Over 65</td>
<td></td>
<td>94608</td>
<td>8</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5147</td>
<td>Over 65</td>
<td></td>
<td>94608</td>
<td>8</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5148</td>
<td>Over 65</td>
<td></td>
<td>94608</td>
<td>8</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5151</td>
<td>Over 65</td>
<td></td>
<td>94608</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>
The Technical Details

RStudio

Retrieves relevant information from clients who have converted from MS Access and runs a regression analysis to determine clients of interest through outliers as well as using Cook’s Distance.

Factors considered: rate of conversion, age of converter, average instructor rating, number of evaluations, number of classes attended
The Technical Details – Cook’s Distance

**RStudio**

\[
D_i = \frac{\sum_{j=1}^{n} (\hat{Y}_j - \hat{Y}_{j(i)})^2}{p \text{ MSE}}.
\]

\[
D_i = \frac{e_i^2}{p \text{ MSE}} \left[ \frac{h_{ii}}{(1 - h_{ii})^2} \right],
\]

\[
D_i = \frac{(\hat{\beta} - \hat{\beta}^{(-i)})^T (X^T X)(\hat{\beta} - \hat{\beta}^{(-i)})}{(1 + p)s^2}.
\]

- \(\hat{Y}_j\) is the prediction from the full regression model for observation \(j\).
- \(\hat{Y}_{j(i)}\) is the prediction for observation \(j\) from a refitted regression model in which observation \(j\) has been omitted.
- \(h_{ii}\) is the \(i\)-th diagonal element of the hat matrix \(X (X^T X)^{-1} X^T\).
- \(e_i\) is the crude residual (i.e., the difference between the observed value and the value fitted by the proposed model).
- \(\text{MSE}\) is the mean square error of the regression model.
- \(p\) is the number of fitted parameters in the model.
How do we price the introductory student special based on seasonal attendance?

**Best Introductory Rate**

- Find Attendance by Month
- Calculate Cost Per Class
- Optimize Introductory Rate
The Technical Details

**[SQL] Average Cost per Class, Average Monthly Attendance**

Calculates:
1. Average cost per class
2. Average monthly attendance for the 30-day window during the intro special

```sql
SELECT Avg(MonthlyData.MonthlyClassAttendance) AS AvgClassAttendance, Avg(MonthlyData.CostPerClass) AS AvgCostPerClass
FROM ((SELECT time, Month, time.Year, Sum(Salary.SalaryPerClass) AS MonthlySalaryCosts,
    m.fc.SumOfAmountPaid AS MonthlyFacilityCosts, (MonthlySalaryCosts + MonthlyFacilityCosts) AS MonthlyCosts,
    ma.MonthlyClassAttendance, (MonthlyCosts * ma.MonthlyClassAttendance) AS CostPerClass
    FROM ((Timeslot AS [time] INNER JOIN
        (SELECT DISTINCT fc.Month, fc.Year, Sum(fc.AmountPaid) AS SumOfAmountPaid
            FROM FacilityCost AS fc GROUP BY fc.Month, fc.Year) AS m.fc ON (time.Year = m.fc.Year)
        AND (time.Month = m.fc.Month)) INNER JOIN MonthlyAttendance AS ma ON (time.Year = ma.Year)
        AND (time.Month = ma.Month)) INNER JOIN ((Instructor AS instruct INNER JOIN
        JOIN Salary ON instruct.InPID = Salary.InPID) INNER JOIN (Class INNER JOIN Teaches AS
tchs ON class.CID = tchs.CID) ON instruct.InPID = tchs.InPID) ON time.TimeID = class.TimeSlotID
    GROUP BY time.Month, time.Year, m.fc.SumOfAmountPaid, ma.MonthlyClassAttendance,
    class.RID

    HAVING ((class.RID=8001)) AS MonthlyData;
```
The Technical Details

SQL: Average Classes Attended & Seasonal Factor

Determines the average number of classes attended during the introductory special.

For each month, it returns seasonal factor (deviation from norm) and the ideal pricing for intro special based on willingness to take 30% cut.

```
SELECT MonthlyAttendance.Month, MonthlyAttendance.Year,
(MonthlyAttendance.MonthlyClassAttendance / AvgCostPerClass.AvgClassAttendance) AS SeasonalFactor,
((SeasonalFactor * AvgClassesAttended.AvgClassesAttended) * AvgCostPerClass.AvgCostPerClass * 0.7)
AS IntroductorySpecialCost
FROM AvgClassesAttended, AvgCostPerClass, MonthlyAttendance;
```

```
SELECT AVG(IntroStudents.ClassesAttended) AS AvgClassesAttended
FROM (
SELECT yp3.[PID-Owns], Count(ac3.CID) AS ClassesAttended
FROM YogaPackage AS yp3 INNER JOIN AttendsClass AS ac3 ON
yp3.PackageID = ac3.PackageID
WHERE (((yp3.PackageName)="Intro Package"))
GROUP BY yp3.[PID-Owns]) AS IntroStudents;
```
## The Technical Details

Access Output

<table>
<thead>
<tr>
<th>Month</th>
<th>Year</th>
<th>SeasonalFactor</th>
<th>IntroductorySpecialCost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2013</td>
<td>0.780462724935733</td>
<td>46.556740474075</td>
</tr>
<tr>
<td>2</td>
<td>2013</td>
<td>0.74879177377892</td>
<td>44.6674814403445</td>
</tr>
<tr>
<td>3</td>
<td>2013</td>
<td>0.816658097686375</td>
<td>48.71589336554814</td>
</tr>
<tr>
<td>4</td>
<td>2013</td>
<td>0.858508997429306</td>
<td>51.2124145214825</td>
</tr>
<tr>
<td>5</td>
<td>2013</td>
<td>1.03835475578406</td>
<td>61.9407068915955</td>
</tr>
<tr>
<td>6</td>
<td>2013</td>
<td>0.867557840616967</td>
<td>51.7522028168341</td>
</tr>
<tr>
<td>7</td>
<td>2013</td>
<td>2.278046272493575</td>
<td>135.891703354764</td>
</tr>
<tr>
<td>8</td>
<td>2013</td>
<td>1.0717223650386</td>
<td>65.4493308133808</td>
</tr>
<tr>
<td>9</td>
<td>2013</td>
<td>0.799691516709512</td>
<td>47.7037906016972</td>
</tr>
<tr>
<td>10</td>
<td>2013</td>
<td>0.894704370179949</td>
<td>53.3715677028889</td>
</tr>
<tr>
<td>11</td>
<td>2013</td>
<td>0.82005143881748</td>
<td>48.9183142662383</td>
</tr>
</tbody>
</table>
Query 4

How do we recommend new instructors to our clients based on their preferences?

Collaborative Filtering

Client Reviews Instructors  Compare Client Preferences  Recommend Top 5
How It Works

[SQL] Retrieves all reviews clients have made in the past towards instructors based on categories

[MS Excel] Assembles the reviews to necessary matrix format to run User-based collaborative filtering

[Recommenderlab] Retrieves Top 5 recommended instructors based on similar user’s ratings and method used
The Technical Details

[SQL]

```
SELECT i.InPID, c.CIPID, ie.Score
FROM Instructor AS i INNER JOIN (Client AS c INNER JOIN InstructorEvaluation AS ie
ON c.CIPID = ie.[Client-PID]) ON i.InPID = ie.[Instructor-PID]
ORDER BY i.InPID;
```

Retrieves instructor evaluations, instructor ID and Customer ID
The Technical Details

[MS Excel]

In order to run UBCF through R, all our client's reviews on every single instructor needs to be in this matrix format.

Export SQL retrieved data into MS Excel and formulate into the appropriate table format.
Assumption is that users with similar preferences will rate instructors similarly.

Missing ratings for a user are predicted by finding a “neighborhood” of similar users and using their ratings to form a prediction.

Measures of similarity between users can be calculated with Pearson's constant (or $R^2$).

```r
> InstructorRecommender = Recommender(sampleyogadata, method = "UBCF")
> # Creates an algorithmic recommender object based on the UBCF system
> # UBCF - User Based Collaborative Filtering
> 
> Predictionslist = predict(InstructorRecommender, sampleyogadata, n=5, type = c("topNList", "ratings"))
> # Generates predictions using the UBCF algorithm to generate a Top 5 Recommended list of Instructors
> 
> as(Predictionslist, "List")

[[1]]
[1] "j01" "j78" "j80" "j89" "j88"

[[2]]
[1] "j93" "j75" "j81" "j87" "j76"
```
Query 5

What are the optimal locations for a new yoga studio?

Facility Location Heatmap

- Locate Highest-Grossing Clients
- Find Client Concentration
- Find Optimal Location
How It Works

[SQL] Query generates the address location and amount of money spent for each client

[Python] Python parses the exported Access text file, geocodes the address to coordinates, and weights each point by spendings

[Google Maps API] Coordinates are passed into an HTML file that generates the heatmap using the Google Maps API
The Technical Details

[SQL]

SELECT p.Address, p.ZIP, p.City, p.State, 
SUM(pr.PurchasePrice+e.PurchasePrice+y.Cost) AS LifetimePurchases 
FROM ((Person AS p INNER JOIN (PackageFinancialTransaction AS pa 
INNER JOIN YogaPackage AS y ON (pa.PackageID = y.PackageID) AND 
(pa.PackageID = y.PackageID) AND (y.PackageID = pa.PackageID)) ON p.PID 
= pa.[PID-Owner]) INNER JOIN EventFinancialTransaction AS e ON p.PID = 
e.[PID-Purchases]) INNER JOIN ProductFinancialTransaction AS pr ON p.PID = pr.[PID-Purchases] 
## The Technical Details

### Access Output

<table>
<thead>
<tr>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>ZIP</th>
<th>LifetimePur</th>
</tr>
</thead>
<tbody>
<tr>
<td>1044 53rd St.</td>
<td>Oakland</td>
<td>CA</td>
<td>94608</td>
<td>$2,804.00</td>
</tr>
<tr>
<td>1333 Park Ave</td>
<td>Emeryville</td>
<td>CA</td>
<td>94608</td>
<td>$2,759.00</td>
</tr>
<tr>
<td>2414 Dana St.</td>
<td>Berkeley</td>
<td>CA</td>
<td>94704</td>
<td>$3,089.00</td>
</tr>
<tr>
<td>2628 Benvenue Ave</td>
<td>Berkeley</td>
<td>CA</td>
<td>94704</td>
<td>$2,359.00</td>
</tr>
<tr>
<td>3 Eton Ct</td>
<td>Berkeley</td>
<td>CA</td>
<td>94705</td>
<td>$2,259.00</td>
</tr>
<tr>
<td>4238 Halleck St</td>
<td>Emeryville</td>
<td>CA</td>
<td>94608</td>
<td>$2,704.00</td>
</tr>
<tr>
<td>4534 Congress Ave</td>
<td>Oakland</td>
<td>CA</td>
<td>94601</td>
<td>$42.00</td>
</tr>
<tr>
<td>5224 Trask St.</td>
<td>Oakland</td>
<td>CA</td>
<td>94601</td>
<td>$2,489.00</td>
</tr>
<tr>
<td>5895 Christie Ave</td>
<td>Emeryville</td>
<td>CA</td>
<td>94608</td>
<td>$115.00</td>
</tr>
</tbody>
</table>
The Technical Details

**Python**

```python
from geopy import geocoders

f = open('listAddresses.txt', 'r')
w = open('cleanAddresses.txt', 'r+w')
x = open('listCoords.txt', 'w')

for line in f:
    l = line.split($
    address = l[0].replace(',,') + \n    weight = l[1].replace('\\n', '')
w.write(address)

for line in w:
    g = geocoders.GoogleV3()
    place, (lat, lng) = g.geocode(line)
w.write("%.5f, %.5f \n" % (lat, lng))
```

Converts file output from Access to list of coordinates

Formats coordinates to be graphed in Google Maps
The Technical Details

[Google Maps API]

\[
\bar{x}_{wc} = \frac{\sum f_i x_i}{\sum f_i} \quad \bar{y}_{wc} = \frac{\sum f_i y_i}{\sum f_i}
\]

where:

\( wc = \text{Weighted Center} \)

\( f = \text{frequency (or weighting factor)} \)
Normalization Analysis

Event (EventID, EventName, DateStarted, DateEnded, TimeStarted, TimeEnded, Cost, RoomID7, FacID32, OtherLocations)

Normalize into 1NF:

- Event (EventID, EventName, DateStarted, DateEnded, TimeStarted, TimeEnded, RoomID7, FacID32)
- OtherLocation (FacID32, DateStarted, DateEnded, TimeStarted, TimeEnded)
- EventCost (RoomID7, FacID32, TimeStarted, TimeEnded, Cost)
Normalization Analysis

Shift (ShiftID, StartTime, EndTime, Day, Hours)

Normalize into 3NF:

- Shift (ShiftID, StartTime, EndTime, Day)
- ShiftLength (StartTime, EndTime, Hours)
Normalization Analysis

Salary(InPID^{15}, SalaryPerClass, OwPID^{13}-Owner, AboveAverageSalary, NumClasses)

Normalize into 2NF, BCNF:

- Salary(InPID^{15}, SalaryPerClass, OwPID^{13}-Owner)
- SalaryStatus(SalaryPerClass, AboveAverage)
- SalaryCalc(NumClasses, SalaryPerClass)
Normalization Analysis

Yoga Package (PackageID, MonthSold, DaySold, YearSold, DateExpires, PackageName, PackageDuration, ClassesAvailable, PID\textsuperscript{10}-Owns, Cost)

Normalize into 1NF, 3NF:

- YogaPackage (PackageID, MonthSold, DaySold, YearSold, PackageName, PackageDuration, PID\textsuperscript{10}-Owns, Cost)
- ClassesAvailable (PackageName, ClassAvailable)
- PackageExpiration (MonthSold, DaySold, YearSold, PackageDuration, DateExpires)
Normalization Analysis

Facility Costs(BillID, FacID, Month, Year, TypeOfCost, HoursUsed, PaidToCompany, AmountPaid)

Normalize into 3NF:

- Facility Costs(BillID, FacID, Month, Year, TypeOfCost, HoursUsed, PaidToCompany)
- TotalCost(TypeOfCost, HoursUsed, Month, AmountPaid)
Thanks for your time!