MARK MULLAN
CATERING EVENTS

Matthew Chiang
Courtney Chow
Tasha Hodgson
Nidhi Kaul
Shir Nehama
Reena Shah
Stephen Sun
Ivan Yung
MEET THE CHEF

Mark Mullan, UC Berkeley senior

Provides local catering & cooking services

Customers: student groups, family, friends
SERVICE OFFERINGS

- Cooking Lessons
- Buffet Catering
- Personal Chef Services
CURRENT OPERATIONS

Company Staff

• Self-operated business
• Up to 4 employees per event

Business Growth

• Customer network is expanding
• Potential for more types of service

Data Organization

• All recorded by hand
• No existing database
POTENTIAL BENEFITS

- Improve customer loyalty, see trends
- Gauge employees’ experiences, personal strengths
- Know exact profit per dish, auto-calculate ingredient amounts, saves time and money
- Better understanding of cash flow
- Track inventory, mitigate excess purchases
RELATIONSHIPS

Microsoft Access Relationship View
RELATIONAL SCHEMA

STRONG ENTITIES
1. Person(PersonID, Fname, Lname, HomePhone#, CellPhone#, Street, City, Zip_Code, Email_Address)
2. Client(ClientID, PersonID, Age, Sex)
3. Supplier(SupplierID, PersonID, Supplier_Name)
4. Attendee(AttendeeID, PersonID, Age, Sex)
5. Employee(EmployeeID, PersonID, Payment - WithdrawalID, Scheduled_at - TimeslotID, Age, Sex, Start_Date, Wage_Per_Hour, Tier_Rank, SpecialtyDishID)
6. Customer(CustomerID, ClientID)
7. Lead(LeadID, ClientID)
8. Company(CompanyID, CustomerID, Company_Name, Description)
9. Individual(IndividualID, CustomerID)
10. Product(ProductID, ProductName, Food_Category, Season, Allergy_Precaution, Description)
11. Transportation(TransportationID, WithdrawalID, Destination, Time_Traveled, Miles_Traveled, Date)
12. Recipe(RecipeID, RecipeName, Recipe_Directions, Servings)
13. Order(OrderID, Has-EventID)
14. Service(ServiceID, Service_Name)
15. Buffet(BuffetID, ServiceID)
16. Personal_Chef(Personal_ChefID, ServiceID)
17. Lesson(LessonID, ServiceID)
18. Inventory(InventoryID, Inventory_Name, Description, Unit, TypeName)
19. Single_Use(Single_UseID, Inventory, Expiration_Date)
20. Reusable(ReusableID, Inventory)
RELATIONAL SCHEMA

**STRONG (cont.)**

21. Task(TaskID, Description)
22. Timeslot(TimeslotID, TimeIn, TimeOut, Date_Date, Date_Day, Date_Month, Date_MonthYear)
23. Marketing(MarketingID, Category, Description, Cost)
24. Financial_Transaction(Financial_TransactionID, Amount, Payment_by_PersonID, Received_by_PersonID, Account_No, Type_of_Account, Date/Time, Description, Comment, EventID)
25. Deposit(DepositID, Financial_TransactionID, Incurs_Payment_From_EventID)
26. Withdrawal(WithdrawalID, Financial_TransactionID)
27. Inventory_Instance(Inventory_InstanceID, InventoryID, Incurs- WithdrawalID, Location_Stored, Expiration_Date)
28. Event(EventID, ServiceID, CustomerID, TimeslotID, Location, Location_City, Location_Zip, Type, Bill_Amount, Amount_Paid, Workers_Needed)
29. Product_Instance(Product_InstanceID, ProductID, Incurs_Payment - Deposit, Expiration_Date, Location_Stored)
30. Feedback(FeedbackID, CustomerID, Rating, Comments)

**WEAK ENTITIES**

43. Skill(EmployeeID, Cooking_Skills)

**MULTI-VALUED ATTRIBUTE**
N:M RELATIONSHIPS

31. Order_ConsistsofProduct(OrderID, Product_InstanceID, ProductID, OrderDate)
32. Inventory_Instance_Usage(RecipeID, Inventory_InstanceID, InventoryID, EventID)
33. Customer_Buys_Product_Instance(CustomerID, Product_InstanceID, ProductID, Date, Quantity)
34. Inventory_Instance_Bought_From_Supplier(Inventory_InstanceID, InventoryID, SupplierID, Date, Cost, Quantity, Units)
35. Transportation_to_Supplier(SupplierID, TransportationID)
36. Transportation_to_Event(EventID, ServiceID, TransportationID)
37. Attends(AttendeeID, EventID, ServiceID, Comments)
38. Performs(EmployeeID, TaskID, EventID)
39. Manages(EmployeeID, Inventory_InstanceID, InventoryID)
40. Marketing_Employment(EmployeeID, MarketingID)
41. Employee_Availability(AvailabilityID, EmployeeID, Avail_Date, Avail_In, Avail_Out)
42. Marketing_Uses(MarketingID, Inventory_InstanceID, InventoryID, Date, Cost)
43. Advertisement(MarketingID, PersonID, Effectiveness)
44. Menu_ItemUsesRecipe(Menu_ItemID, RecipeID)
45. Order_ConsistsofMenuItem(Menu_ItemID, OrderID, OrderQuantity)
46. Product_UsesRecipe(ProductID, RecipeID)
47. Employee_Scheduled_At(SchedID, EventID, EmployeeID)
48. RecipeUsesInventory(RecipeUsesInventoryID, RecieptID, InventoryID, InventoryName, Quantity, Unit)
QUERIES
QUERY 1: EVENT OPTIMIZATION

For particular event feedback, how can Mark assess which events, employees, or people are affected?

- Feedback forms given to clients/attendees
- Mark will assess areas of improvement
- Employee enhancement
- Food preferences
- Optimize customer satisfaction
Business Justification

- Track whether there is an association between low ratings and specific aspects of the event
- Optimize customer satisfaction by pinpointing the root cause of negative feedback
- Identify areas where employees need more training
**QUERY 1: EVENT OPTIMIZATION**

### Order Feedback

```sql
```

### Employee Feedback

```sql
```
QUERY 1: EVENT OPTIMIZATION

SQL CODE

Service Feedback

Seasonality Feedback
FROM Timeslot INNER JOIN (Event INNER JOIN Feedback ON Event.EventID = Feedback.EventID) ON Timeslot.TimeslotID = Event.TimeslotID;
**QUERY 1: EVENT OPTIMIZATION**

**Sample Output**

<table>
<thead>
<tr>
<th>ServiceID</th>
<th>Service_Name</th>
<th>Friendliness</th>
<th>Scheduling</th>
<th>Timeliness</th>
<th>FoodQuality</th>
<th>FoodQuantity</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Buffet</td>
<td>Buffet</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>1 Buffet</td>
<td>Buffet</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>2 Personal Chef</td>
<td>2 Personal Chef</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2 Personal Chef</td>
<td>2 Personal Chef</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>2 Personal Chef</td>
<td>2 Personal Chef</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>3 Lesson</td>
<td>Lesson</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3 Lesson</td>
<td>Lesson</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

*Friendliness, Timeliness, FoodQuality, etc. ranked on scale of 1-5*
QUERY 1: EVENT OPTIMIZATION

Boxplots of Feedback

Step 1: Normalize feedback scores using

\[
\frac{x_i - \min(x)}{\text{Range}(x)}
\]

Step 2: Create box plots to visualize spread of each feedback category

R CODE

```r
```
QUERY 1: EVENT OPTIMIZATION

- Distribution of Standardized Friendliness Rating
- Distribution of Standardized Scheduling Rating
- Distribution of Standardized Timeliness Rating
- Distribution of Standardized Food Quality Rating
- Distribution of Standardized Food Quantity Rating
- Distribution of Standardized Overall Rating
**Chi-Squared Analysis**

**R Code of a Sample**

```r
order.quality.frame = data.frame(order[1], order[5])
order.quality = table(order.quality.frame)
chisq.test(order.quality)
```

**Chi-Squared Equation**

\[ \chi^2 = \sum \frac{(O - E)^2}{E} \]

**Hypothesis Testing**

- **H\(_0\)**: No Association
- **H\(_a\)**: There exists an association
  - \(\alpha = 0.1\) yields results with 90% confidence
  - If p-value < 0.1, the variables are statistically significantly dependent

**P-Values of Analysis**

<table>
<thead>
<tr>
<th>Analysis</th>
<th>p-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Food Order versus Food Quality Rating</td>
<td>p = 0.2548</td>
</tr>
<tr>
<td>2) Food Order versus Food Quantity Rating</td>
<td>p = 0.2395</td>
</tr>
<tr>
<td>3) Food Order versus Overall Rating</td>
<td>p = 0.7596</td>
</tr>
<tr>
<td>4) Employee versus Friendliness</td>
<td>p = 0.3353</td>
</tr>
<tr>
<td>5) Employee versus Overall Rating</td>
<td>p = 0.7968</td>
</tr>
<tr>
<td>6) Service versus Friendliness Rating</td>
<td>p = 0.4232</td>
</tr>
<tr>
<td>7) Service versus Scheduling Rating</td>
<td>p = 0.1557</td>
</tr>
<tr>
<td>8) Service versus Food Quantity Rating</td>
<td>p = 0.3869</td>
</tr>
<tr>
<td>9) Service versus Overall Rating</td>
<td>p = 0.07489</td>
</tr>
<tr>
<td>10) Season versus Scheduling Rating</td>
<td>p = 0.47</td>
</tr>
<tr>
<td>11) Season versus Timeliness Rating</td>
<td>p = 0.22</td>
</tr>
<tr>
<td>12) Season versus Food Quality Rating</td>
<td>p = 0.715</td>
</tr>
<tr>
<td>13) Season versus Overall Rating</td>
<td>p = 0.567</td>
</tr>
</tbody>
</table>
What are the ingredients required for a specific order? What is the inventory stock level after an order is executed?

- Mark speaks with client
- Finalizes catering orders
- Books event date
- Checks inventory stock levels after order
- Checks required ingredients for order
Caters business runs based on orders received from customers

Inventory levels constantly adjust whenever purchases are made and customer orders are executed

Enter new order, check required ingredients, check inventory stock levels left after execution of order
You have a new order!

Order 9 consists of several Menu items of varying quantities.

Order 9 consists of 10 Salmon and Risotto menu items

Each Menu item may have 1+ associated recipes.

This Menu item requires: Salmon recipe, Risotto recipe

Each Recipe may require 1+ ingredients.

Salmon recipe: 3 oz. salmon, 1 pint chicken stock, 2 tbs. oil
Risotto recipe: 2 oz. Risotto rice, 1 tbs. shredded cheese
**Part A: Get quantity of inventory item for specific order**

SELECT Order_ConsistsofMenuItem.OrderID, Menu_Item.Menu_ItemID, Menu_ItemUsesRecipe.Recipe_ID, RecipeUsesInventory.InventoryID, RecipeUsesInventory.InventoryName, Order_ConsistsofMenuItem.Quantity*RecipeUsesInventory.Quantity AS Quantity_Needed, RecipeUsesInventory.Unit

FROM ((Menu_Item INNER JOIN Order_ConsistsofMenuItem ON Menu_Item.Menu_ItemID = Order_ConsistsofMenuItem.Menu_ItemID) INNER JOIN Menu_ItemUsesRecipe ON Menu_Item.Menu_ItemID = Menu_ItemUsesRecipe.Menu_ItemID) INNER JOIN RecipeUsesInventory ON Menu_ItemUsesRecipe.Recipe_ID = RecipeUsesInventory.RecipeID

Now, aggregate inventory items that appear multiple times within one order.

**SQL Code**

```sql
SELECT [2_1 Order's Ingredients].OrderID, [2_1 Order's Ingredients].InventoryID, Sum([2_1 Order's Ingredients].Quantity_Needed) AS SumOfQuantity_Needed, [2_1 Order's Ingredients].Unit
FROM [2_1 Order's Ingredients]
GROUP BY [2_1 Order's Ingredients].OrderID, [2_1 Order's Ingredients].InventoryID, [2_1 Order's Ingredients].Unit;
```
Part B: We need a series of queries that will update inventory stock levels.

**#1.**
First, we group by the inventory item and sum their quantity across all orders in order to see the total amount of the inventory quantity used in our orders.

<table>
<thead>
<tr>
<th>InventoryID</th>
<th>Total_Inventory_Used</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>126 ounces</td>
<td>ounces</td>
</tr>
<tr>
<td>15</td>
<td>42 pint</td>
<td>pint</td>
</tr>
<tr>
<td>16</td>
<td>42 tablespoon</td>
<td>tablespoon</td>
</tr>
</tbody>
</table>

**#2.**
Then, we also want to see how much of an inventory instance (for each inventory) we have bought.

<table>
<thead>
<tr>
<th>InventoryID</th>
<th>Total_Inventory_Bought_Quantity</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>1035 ounces</td>
<td>ounces</td>
</tr>
<tr>
<td>15</td>
<td>2000 pint</td>
<td>pint</td>
</tr>
<tr>
<td>16</td>
<td>1500 tablespoon</td>
<td>tablespoon</td>
</tr>
</tbody>
</table>

**#2 minus #1.**
Find **Total Inventory quantity in stock** by subtracting the total inventory used by our orders from total inventory bought.

<table>
<thead>
<tr>
<th>InventoryID</th>
<th>Total_Inventory_In_Stock</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>909 ounces</td>
<td>ounces</td>
</tr>
<tr>
<td>15</td>
<td>1958 pint</td>
<td>pint</td>
</tr>
<tr>
<td>16</td>
<td>1458 tablespoon</td>
<td>tablespoon</td>
</tr>
</tbody>
</table>

This query keeps track of how the inventory stock levels change as new orders come in and new inventory is bought.
QUERY 2: ORDER & INVENTORY STOCK

Microsoft Access: Forms + Reports

With ability to track inventory levels & associated ingredient names/quantities, Mark can track exact quantities of inventory he either needs for orders or currently has in storage.

1. Add order to Order table

2. Add order’s menu item dishes + their quantities

3. Check all the ingredients (inventory items) required to complete order

4. Check inventory stock levels as a result of added order
QUERY 2: ORDER & INVENTORY STOCK

Microsoft Access: Forms + Reports
QUERY 2: ORDER & INVENTORY STOCK

Inventory Required For Orders

<table>
<thead>
<tr>
<th>Order ID</th>
<th>Inventory ID</th>
<th>Inventory Name</th>
<th>Quantity Needed</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14</td>
<td>Salmon</td>
<td>30</td>
<td>ounces</td>
</tr>
<tr>
<td>15</td>
<td>Oil</td>
<td>10</td>
<td>tablespoon</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Butter</td>
<td>110</td>
<td>tablespoon</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Onion</td>
<td>700</td>
<td>grams</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Risotto Rice</td>
<td>20</td>
<td>ounces</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Parmesan Cheese</td>
<td>10</td>
<td>tablespoon</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Tomato</td>
<td>500</td>
<td>grams</td>
<td></td>
</tr>
</tbody>
</table>

Inventory Stock Level

<table>
<thead>
<tr>
<th>Inventory ID</th>
<th>Inventory Name</th>
<th>Stock Level</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Salmon</td>
<td>849</td>
<td>ounces</td>
</tr>
<tr>
<td>15</td>
<td>Chicken Stock</td>
<td>1938</td>
<td>pint</td>
</tr>
<tr>
<td>16</td>
<td>Oil</td>
<td>1438</td>
<td>tablespoon</td>
</tr>
<tr>
<td>17</td>
<td>Butter</td>
<td>1286</td>
<td>tablespoon</td>
</tr>
<tr>
<td>18</td>
<td>Onion</td>
<td>75660</td>
<td>gram</td>
</tr>
<tr>
<td>19</td>
<td>Risotto Rice</td>
<td>876</td>
<td>ounces</td>
</tr>
<tr>
<td>20</td>
<td>Parmesan Cheese</td>
<td>130</td>
<td>tablespoon</td>
</tr>
<tr>
<td>21</td>
<td>Tomato</td>
<td>-124</td>
<td>grams</td>
</tr>
<tr>
<td>22</td>
<td>Potato</td>
<td>146000</td>
<td>grams</td>
</tr>
</tbody>
</table>
What is the most cost-efficient way to allocate available employees to upcoming events?

- Mark books event date
- Certain employees available per event
- Assign employee to events
Business Justification

- As business grows, number of events and employees increase – increasing necessity for cost-efficient delegation
- Client can benefit from linear program to find most cost-effective ratio of employees to events
- Will minimize costs and ensure that enough employees will be available per event
QUERY 3: EMPLOYEE SCHEDULE

SQL extracts employees’ available time slots and event times > CSV file.

1. SELECT E.EventID, T.Date_Date, T.Time_In, T.Time_out, E.Workers_Needed
   FROM Timeslot AS T INNER JOIN Event AS E ON T.TimeslotID = E.TimeslotID;

2. SELECT E.EmployeeID, A.Avail_Date, A.Avail_In, A.Avail_Out, E.Wage_Per_Hour
   FROM Employee_Availability AS A INNER JOIN Employee AS E ON A.EmployeeID = E.EmployeeID;
Python code finds all employees available to work for an event, by parsing and matching time slots. Creates .dat file.

```python
import csv
import re

# Function to check employee availability
def isempavail(etimemin, etimemax, emtimemin, emtimemax):
    etimemin_split = etimemin.split(" ")
    etimemax_split = etimemax.split(" ")
    emtimemin_split = emtimemin.split(" ")
    emtimemax_split = emtimemax.split(" ")
    if etimemin_split[0] == etimemax_split[0]:
        if etimemin_split[1] == etimemax_split[1]:
            if etimemin_split[2] == etimemax_split[2]:
                if emtimemin_split[0] == emtimemax_split[0]:
                    if emtimemin_split[1] == emtimemax_split[1]:
                        if emtimemin_split[2] == emtimemax_split[2]:
                            return True
    return False

# Read EventTimes
with open('eventTimes.csv') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ', quotechar='')
    EventTimes = []

    for eventid, date, timemin, timemax in reader:
        date, timemin, timemax = date.split(' ', 2)
        if eventid in EventTimes.keys():
            EventTimes[eventid].append((date, timemin, timemax))
        else:
            EventTimes[eventid] = [(date, timemin, timemax)]

# Read Employee Availability
with open('employeeAvailability.csv') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ', quotechar='')
    EmpAvail = []

    for key, empid, date, timemin, timemax in reader:
        date, timemin, timemax = date.split(' ', 2)
        if date in EmpAvail.keys():
            EmpAvail[date].append((empid, timemin, timemax))
        else:
            EmpAvail[date] = [(empid, timemin, timemax)]

# Main Logic
final = []
keys = EventTimes.keys()
keys.sort()

for eventid in keys:
    edate = EventTimes[eventid][0][0]
etimein = EventTimes[eventid][0][1]
etimemax = EventTimes[eventid][0][2]

    if isempavail(etimein, edate, timemax, emtimemin, emtimemax):
        for availrecord in EmpAvail[edate]:
            empid = availrecord[0]
etimein = availrecord[1]
etimemax = availrecord[2]

            if empid not in temp:
                temp.append(empid)

        # Print EventTimes
        print EventTimes

# Write to CSV file
with open('query3.csv', 'w') as csvfile:
    writer = csv.writer(csvfile, delimiter=' ', quotechar='')
    for key in final:
        writer.writerow(key)

print key
```
LINEAR PROGRAM

AMPL takes in .mod and .dat file. Mod file created beforehand. Python hands .dat file to AMPL. AMPL solves and returns solution.

PYTHON

Python takes in the solutions file and manipulates it to be pushed back into MS Access.
QUERY 3: EMPLOYEE SCHEDULE

Sample Output

Can check EventID to see which employees are assigned to work on that day.
After his company expands, how can Mark efficiently allocate tasks amongst many available employees?

1. Export data from Access to SQL Server using MSA Upsizing Wizard, choosing those tables which relate to employee availability, new events, and required tasks.

2. Create Asana account: allows for online Workspaces, Projects, Personal Projects, Tasks, Tags, Notes, Comments, and Inbox to communicate effectively within large teams.


4. Now, each time Asana senses a SQL trigger (and vice versa), the other will update itself automatically.
QUERY 3: EMPLOYEE SCHEDULE

Employee Schedule & Task Optimization

Triggers

- New Column
  - Triggered when you add a new column.

- New Table
  - Triggered when you add a new table.

- New Row (Custom Query)
  - Triggered when new rows are returned from a custom query that you provide. **Advanced Users Only**

- New Row
  - Triggered when you add a new row.

Actions

- Create Project
  - Adds a new project.

- Create Story
  - Adds a new story.

- Create Task
  - Adds a new task.

a: Triggers

- New Project
  - Triggered when you add a new project.

- New Comment
  - Triggered when you add a new comment.

- New Story
  - Triggered when you add a new story.

- New Tag
  - Triggered when you add a new tag.

- New User
  - Triggered when you add a new user.

- New Workspace/Organization
  - Triggered when you add a new workspace/organization.

- New Team
  - Triggered when you add a new team.

- New Task
  - Triggered when you add a new task.

Actions
How can we assess the financial health of the business + identify opportunities and projections of future growth?
Business Justification

- Query obtains information on current financial health of business, in addition to future growth potential
- Discover which performed services are most lucrative, leading to smarter business decisions and modifications
- Indicates beneficial and detrimental actions by extracting relevant data (revenue, cost & total profit for events per month)
QUERY 4: FINANCIAL ANALYSIS

**SQL CODE**

**For Service-Specific Data**

SELECT SUM(D.Total_Revenue) AS Revenue, SUM(W.Total_Withdrawal) AS Cost, Revenue-Cost AS Profit, T.Date_MonthYear
WHERE S.Service_Name = 'Buffet'
GROUP BY T.Date_MonthYear;

**For Total Profits/Revenues + Costs Per Event + Calculating Profit Margin**

SELECT Sum(D.Total_Revenue) AS Revenue, Sum(W.Total_Withdrawal) AS Cost, Revenue-Cost AS Profit, Profit/Revenue AS Profit_Margin, T.Date_MonthYear
FROM Timeslot AS T INNER JOIN (((4_1_DepositsTransactions AS D INNER JOIN 4_8_WithdrawalTransactions AS W ON D.EventID = W.EventID) INNER JOIN Event AS E ON E.EventID = W.EventID) INNER JOIN Service AS S ON E.ServiceID = S.ServiceID) ON T.TimeslotID = E.TimeslotID
WHERE S.Service_Name = 'Buffet'
GROUP BY T.Date_MonthYear;
### QUERY 4: FINANCIAL ANALYSIS

#### Sample Outputs

<table>
<thead>
<tr>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>$208.00</td>
<td>$90.00</td>
<td>$118.00</td>
<td>August 2013</td>
</tr>
<tr>
<td>$195.00</td>
<td>$88.00</td>
<td>$107.00</td>
<td>December 2013</td>
</tr>
<tr>
<td>$160.00</td>
<td>$84.00</td>
<td>$76.00</td>
<td>March 2013</td>
</tr>
<tr>
<td>$246.00</td>
<td>$131.00</td>
<td>$115.00</td>
<td>May 2013</td>
</tr>
</tbody>
</table>

Profits from Cooking Lessons

<table>
<thead>
<tr>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>$230.00</td>
<td>$119.39</td>
<td>$110.61</td>
<td>April 2013</td>
</tr>
<tr>
<td>$307.45</td>
<td>$39.36</td>
<td>$268.09</td>
<td>February 2013</td>
</tr>
<tr>
<td>$200.00</td>
<td>$83.00</td>
<td>$117.00</td>
<td>June 2013</td>
</tr>
<tr>
<td>$240.00</td>
<td>$85.00</td>
<td>$155.00</td>
<td>November 2013</td>
</tr>
<tr>
<td>$308.00</td>
<td>$208.00</td>
<td>$100.00</td>
<td>September 2013</td>
</tr>
</tbody>
</table>

Profits from Personal Chef Service

<table>
<thead>
<tr>
<th>Revenue</th>
<th>Cost</th>
<th>Profit</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>$523.43</td>
<td>$141.97</td>
<td>$381.46</td>
<td>January 2013</td>
</tr>
<tr>
<td>$480.00</td>
<td>$326.00</td>
<td>$154.00</td>
<td>May 2013</td>
</tr>
<tr>
<td>$356.00</td>
<td>$182.00</td>
<td>$174.00</td>
<td>October 2013</td>
</tr>
<tr>
<td>$372.00</td>
<td>$194.00</td>
<td>$178.00</td>
<td>November 2013</td>
</tr>
</tbody>
</table>

Profits from Buffet-Style Catering

<table>
<thead>
<tr>
<th>Revenue</th>
<th>Profit</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>$110.61</td>
<td>0.4890130434378261 April 2013</td>
<td></td>
</tr>
<tr>
<td>$118.00</td>
<td>0.5673076923076922 August 2013</td>
<td></td>
</tr>
<tr>
<td>$107.00</td>
<td>0.5487179481794819 December 2013</td>
<td></td>
</tr>
<tr>
<td>$268.09</td>
<td>0.87197183360709 February 2013</td>
<td></td>
</tr>
<tr>
<td>$381.46</td>
<td>0.72876984506467 January 2013</td>
<td></td>
</tr>
<tr>
<td>$326.00</td>
<td>0.3208333333333333 July 2013</td>
<td></td>
</tr>
<tr>
<td>$154.00</td>
<td>0.585 June 2013</td>
<td></td>
</tr>
<tr>
<td>$76.00</td>
<td>0.47849423655914 October 2013</td>
<td></td>
</tr>
<tr>
<td>$100.00</td>
<td>0.324657324675325 September 2013</td>
<td></td>
</tr>
</tbody>
</table>

Net Profits
QUERY 4: FINANCIAL ANALYSIS

Microsoft Access: Form

**Mark Mullan Events**
FINANCIAL ANALYSIS

**PROFIT TOTALS**
- See Buffet Profits
- See Lesson Profits
- See Personal Chef Profits
- See Overall Profits

**FOR FURTHER ANALYSIS**
- Export Buffet Profits into Excel
- Export Lesson Profits into Excel
- Export Personal Chef Profits into Excel
- Export Overall Profits into Excel

Allows client to quickly access & view queries that pull data with various profit totals.

Can export the queries straight into Excel to graph data and perform analyses on it.
Graphs of Moving Average Regressions

Sample application of data (very small sample size)
Graphs of Moving Average Regressions

Sample application of data (very small sample size)
How can we determine the optimal price for each of Mark’s events?
Business Justification

- Analyzes how certain costs and events compare to factors such as seasonality, location, and skills required
- Provides insight and analysis into core of how his business generates revenue, as well as forecasting functionality
QUERY 5: OPTIMAL PRICING

Multiple Regression Analysis / ANOVA

Optimal Pricing Model

\[-93.06 + 3.59 \times \text{(Labor Cost)} + 3.01 \times \text{(Product Cost)} - 3.89 \times \text{(Transportation Cost)} + 20.29 \times \text{(Berkeley)} + 104.66 \times \text{(San Francisco)} + 0.01 \times \text{(San Jose)} + 3.24 \times \text{(Fall)} + 70.76 \times \text{(Winter)} + 164.58 \times \text{(Spring)} + 0.02 \times \text{(Summer)} - 27.52 \times \text{(Buffet)} - 38.92 \times \text{(Personal Chef)} + 0.02 \times \text{(Lesson)} + 51.38 \times \text{(Skills)}\]

Via Microsoft Excel

\[R^2 = 0.99\]

Adjusted \(R^2 = 0.75\)
QUERY 5: OPTIMAL PRICING

SQL CODE

**Labor Cost**

```
SELECT Event.EventID, Employee_Scheduled_At.TimeslotID,
Sum([employee.wage_per_hour]*[employee_scheduled_at.hours_worked]) AS Total
FROM Employee INNER JOIN (Event INNER JOIN Employee_Scheduled_At ON Event.TimeslotID = 
Employee_Scheduled_At.TimeslotID) ON Employee.EmployeeID = Employee_Scheduled_At.EmployeeID
GROUP BY Event.EventID, Employee_Scheduled_At.TimeslotID;
```

**Product Cost**

```
SELECT Order.EventID, Sum([order_consistsofmenuitem.quantity]*[product.cost]) AS Total
FROM (Product INNER JOIN ([Order] INNER JOIN Order_ConsistsofProduct ON Order.OrderID = 
Order_ConsistsofMenuItem ON Order.OrderID = Order_ConsistsofMenuItem.OrderID
GROUP BY Order.EventID, Order.OrderID;
```
QUERY 5: OPTIMAL PRICING

**Transportation Cost**

```sql
SELECT Transportation_to_Event.EventID, Sum(Financial_Transaction.Amount) AS SumOfAmount
GROUP BY Transportation_to_Event.EventID;
```

**Location**

```sql
SELECT Event.EventID, Event.Location, Event.Location_City, Event.Location_Zip
FROM Event;
```

**Seasonality**

```sql
SELECT Event.EventID, Timeslot.Date_Month
FROM Timeslot INNER JOIN Event ON Timeslot.TimeslotID = Event.TimeslotID;
```
QUERY 5: OPTIMAL PRICING

**SQL CODE**

**Service**

```sql
SELECT Event.EventID, Event.ServiceID, Service.Service_Name
FROM Service INNER JOIN Event ON Service.ServiceID = Event.ServiceID;
```

**Skills**

```sql
FROM (Person INNER JOIN (Timeslot INNER JOIN (Employee INNER JOIN Skill ON Employee.EmployeeID = Skill.EmployeeID) ON Timeslot.TimeslotID = Employee.Scheduled_At_Timeslot) ON Person.PersonID = Employee.PersonID) INNER JOIN Event ON Timeslot.TimeslotID = Event.TimeslotID;
```

**Number of Skills**

```sql
SELECT [5_2_skills].[eventID],
       COUNT ([5_2_skills].[Cooking_Skill]) AS NumOfSkills
FROM [5_2_skills]
GROUP BY [5_2_skills].[eventID];
```
SELECT DISTINCT Event.EventID,  -93.06+3.59*([5_1_laborcost].[Total])+
3.01*([5_1_productcost].[Total])-3.89*([5_1_transportationcost].[SumOfAmount])
+(IIf([5_2_location].[Location_City]="Berkeley",+20.29,
IIf([5_2_location].[Location_City]="San Francisco",+104.66,
IIf([5_2_location].[Location_City]="San Jose",+0.01,
IIf([5_2_seasonality].[Date_Month]="September" OR
[5_2_seasonality].[Date_Month]="October" OR
[5_2_seasonality].[Date_Month]="November",+3.24,
IIf([5_2_seasonality].[Date_Month]="December" OR
[5_2_seasonality].[Date_Month]="January" OR
[5_2_seasonality].[Date_Month]="February",+70.76,
IIf([5_2_seasonality].[Date_Month]="March" OR
[5_2_seasonality].[Date_Month]="April" OR
[5_2_seasonality].[Date_Month]="May",+164.58,
IIf([5_2_seasonality].[Date_Month]="June" OR
[5_2_seasonality].[Date_Month]="July" OR
[5_2_seasonality].[Date_Month]="August",+0.02,
IIf([5_2_service].[Service_Name]="Buffet",-27.52,
IIf([5_2_service].[Service_Name]="Personal Chef",-38.92,
IIf([5_2_service].[Service_Name]="Lesson",+0.02,
IIf([5_2_countskills].[NumOfSkills]>0,+51.38)))))))))))))) AS ProjectedPrice
FROM Event, 5_1_laborcost, 5_1_productcost,
5_1_transportationcost, 5_2_location, 5_2_seasonality,
5_2_service, 5_2_countskills
WHERE ((([5_1_laborcost].EventID)=[Event].[EventID])
AND ((([5_1_productcost].EventID)=[Event].[EventID])
AND ((([5_1_transportationcost].EventID)=[Event].[EventID])
AND ((([5_2_location].EventID)=[Event].[EventID])
AND ((([5_2_seasonality].EventID)=[Event].[EventID])
AND ((([5_2_service].EventID)=[Event].[EventID])
AND ((([5_2_countskills].EventID)=[Event].[EventID]))) AS ProjectedPrice
QUERY 5: OPTIMAL PRICING

Sample Output

- Outputs optimal event price based on Excel multiple regression analysis
- Uses IFF statements to toggle various event details
- Obtains required variables
NORMALIZATION
NORMALIZATION: 1NF

Product (ProductID, ProductName, Food_Category, Season, Allergy_Precaution)

Remove Multi-Valued Attributes:

1. Product (ProductID, ProductName, Food_Category)
2. Season (ProductID, Season)
3. Allergy_Precaution(ProductID, AllergyID)
NORMALIZATION: 2NF

Remove Partial Key Dependency:

* Pi1 (Product_Instance_ID, Product_ID, Deposit, Expiration)
* Pi2 (Product_ID, Location_Stored)
NORMALIZATION: 3NF

Employee (Employee_ID, Person_ID, Payment_W ithdrawal_ID, Scheduled_at_TimeslotID, Age, Sex, Start_Date, Wage, Tier)

Remove Transitive Dependencies:
E1 (Employee_ID, Person_ID)
E2 (Employee_ID, Payment_W ithdrawal_ID, Scheduled_at_TimeslotID, Start_Date, Tier)
E3 (Tier, Wage)
E4 (Person_ID, Age, Sex)

(Already in BCNF)
THANK YOU