Team 6

Anjali Gopal
Neil Garde
Sirisha Varigonda
Ka Hei Cheung

Fannie Huang
Burt Chen
Yan Zhuang
Huanyi Zhang

Anirudh Kilambi
Peter Wysinski
Company Overview

Website: berkeleydelivers.com

Service Offerings:
Delivers meals from local restaurants
Expanding to fulfill other deliveries

Employees:
5 drivers & 2 dispatchers

Overview:
~120 orders a night (75% orders online, 25% orders via phone)
34 restaurants
Hungry?
Meals delivered from $7
Order from your favorites below:
Order Up!

We're open every day from 5pm - 11pm

How It Works

Ordering food should be convenient and affordable. Feeling under the weather? Under a time crunch to get that paper done? Just choose your favorite dish and we'll take care of the rest. We bring the restaurant to your home, all you need to do is eat.

Once you place your order, we'll send a text message to the number provided with a link that allows you to track your meal. You'll be able to know as your order is being prepared, and can prepare your hunger knowing that your driver is on his way.

Order

Sit Tight

Bon Appétit
Gypsies Italian

Chicken Plates

Pollo Con Pesto
$9 - Chicken, Pesto sauce, Parmigiano, trace of cream tossed w/ Fettuccine

Chicken Parmigiano
$9 - Chicken breast with marinara sauce & Mozzarella cheese over Spaghetti

Melanzana Con Pollo
$9 - Chicken, Spinach, Gilled eggplant, Garlic, Tomato sauce, tossed w/ Penne

Chicken Gnocchi
$9 - Tricolor potato pasta, Chicken, and cream sauce

Crazy Alfredo
$9 - Chicken, Sausage, Roasted red peppers, Red wine, in Alfredo sauce tossed w/ Fettuccine

Chicken and Broccoli
$9 - Chicken, Broccoli, Garlic, Parmigiano cheese tossed w/ Penne
Developed System

Designed specifically to provide insight

Optimizes the dispatching process

Assesses efficiency of operation
✓ for the restaurant
✓ for the driver

Deals with coupons & advertisements
Access Relationships
Relational Schema
Relational Schema

STRONG ENTITIES:

1. Customer (Customer_ID, FName, Lname, Phone, Email_Address, Age, Street\(^8\), City\(^8\), State\(^8\), Country\(^8\))
2. Advertisement (AID, Advertise_Type, Revenue growth, Referral, Price, Location)
3. Coupon (Coupon_code, Discount_Percentage, Business_ID\(^6\))
4. Order (OID, Customer_ID\(^1\), Order_Type, Coupon_Code\(^3\), SSN\(^10\), Delivering_Time, ZIPCode, Ordering_Time, Delivering_Distance, Employee_Num\(^10\), Driver_license_Num\(^{10.2}\))
   4.1. Online (OID\(^4\), IP_Address, Payment_No.\(^5\))
   4.2. Fax (OID\(^4\), Fax_Number, Payment_No.\(^5\))
   4.3. Phone (OID\(^4\), Phone_Number, Payment_No.\(^5\))
5. Business (Business_ID, Business_Name, Business_Type, ZIPCode, Street\(^7\), City\(^7\), State\(^7\), Country\(^7\), SSN\(^9\), Employee_Num\(^9.1\), Reimbursement_amount)
   5.1. Restaurant (Business_ID\(^5\), Restaurant_Name, Food_Type, Rating, Price_level)
   5.2. Convenience Store (Business_ID\(^5\), Product_type, Rating)
Relational Schema

STRONG ENTITIES (Cont.):
6. Product (PID, Business_ID, Product_Name, Product_Price, Product_Ingredients, Allergens_Warnings)
7. Location (Street, City, State, Country)
8. Feedback (Feedback_ID, Feedback_type, Customer_ID)
   8.1. Other (Feedback_ID, SSN, Employee_Num)
   8.2. Delivery Feedback (Feedback_ID, Order_Type, Delivery_Speed, Driver_Friendliness, Food_Quality)
9. Employee (SSN, Employee_Num, Product_ID, Salary, Birthday, Fname, Lname, Email, Job_Position, Phone_Num, Banking_Info, Managed_Department, Department, Working_Schedule, Qty_Hours_Worked)
   9.1. Manager (SSN, Employee_Num, Managed_Department)
   9.2. Driver (SSN, Employee_Num, Driver_license_Num, Vehicle_type, Vehicle_License_plate, Phone_Number, Driving_Distance)
   9.3. Dispatcher (SSN, Employee_Num, Phone_Number)
Relational Schema

WEAK ENTITIES:
10. Payment (Payment_No., Customer_ID₁, Payment_type, Receipt, Time, Date)
   10.1. Credit Card (Payment_No.¹⁰, Card_Num, Card_type, Name_on_card, Expire_date,
         Security_code, SSN⁹, Employee_Num⁹, Managed_Department⁹.₁)
   10.2. Cash (Payment_No.¹⁰, SSN⁹, Employee_Num⁹, Driver_license_Num⁹.₂)

MULTI-VALUED ATTRIBUTES:
11. Zipcodes (Street⁷, City⁷, State⁷, Country⁷, Zipcode)
12. Product_types (PID, Product_Type)
13. Emails (Customer_ID₁, Email_address)
N:M RELATIONSHIPS:

14. Targets (Customer_ID¹, AID², Advertisement_Costs)
15. Assigns (SSN⁹¹, Employee_Num⁹¹, Driver_license_Num⁹¹, SSN⁹², Employee_Num⁹²)
16. Contains (Coupon_code³, AID²)
17. Are_Featured (Business_ID⁵, AID²)
18. Pays (SSN⁹, Employee_Num⁹, Driver_license_Num⁹², Business_ID⁵)
Queries
Query 1 - Timeslot Demand

How many orders are placed on a given hour of the day for a given location / business?

Procedure:
Determine the demand on different time slots via a count of number of order placed.

Utility:
✓ Staffing: Forecast the demand and staff an appropriate number of drivers and dispatchers at different times of the day
✓ Capital allocation: Knowledge of increased expected demand can be used to rent/acquire capital resources
Query 1 - Timeslot Demand

**SQL 1 - Number of orders placed on a given timeslot for a given business**

```sql
SELECT b.Business_ID, HOUR(o.Delivering_Time) AS Hour, count(*) AS NumOfOrder
FROM Order AS o, Business AS b, Product AS p
GROUP BY o.Business_ID, HOUR(o.Time)
ORDER BY count(*) DESC;
```

**SQL 2 - Number of orders placed on a given timeslot for a given location**

```sql
SELECT b.ZIPCode, HOUR(o.Delivering_Time) AS Hour, count(*) AS NumOfOrder
FROM Order AS o, Business AS b, Product AS p
GROUP BY o.ZIPCode, HOUR(o.Time)
ORDER BY count(*) DESC;
```
Query 1 - Timeslot Demand

Linear Programming - AMPL

We use AMPL to find the most cost-efficient method of arranging the drivers in order to meet demands at specific time-slots.
Query 2 - Driver Analysis/Evaluation

What is the average delivery time of a driver in a given city?

Procedure:
Determine which drivers are efficient or inefficient based on the average time used to complete orders and number of orders completed

Utility:
✓ Performance metrics: Can be used to evaluate driver performance and give feedback to underperforming drivers
✓ Customer experience: Can ascertain where or why orders were delayed, and to improve consumer experiences
SQL - Average time used to deliver orders and the number of orders completed

```
SELECT b.City, b.Employee_Num, AVG(HOUR(o.Delivering_Time)*60+MINUTE(o.Delivering_Time) - HOUR(o.Ordering_Time)*60+MINUTE(o.Ordering_Time)) AS Avg_Deliver_Time, count(*) AS Order_Delivered
FROM Order AS o, Business AS b, Product AS p
GROUP BY b.City, b.Employee_Num
ORDER BY count(*) DESC;
```
Extract drivers’ delivery time for each city and make a boxplot to see the distribution of the data.
Boxplot for Drivers’ delivery time for 5 cities: Berkeley, Oakland, Richmond, Albany, Emeryville.
Query 3 - Clustering of Restaurant Preferences

Is a preference for one restaurant correlated with preference for another?

Procedure:
Gather customer order data to find the most likely additional preference given the available information

Utility:
✓ Suggestion system: can predict and suggest new restaurants to a user given his or her restaurant preferences
✓ Advertising optimization: advertisements for specific restaurants can be targeted to users who are more likely to purchase given their previous history
SELECT o1.Order_Type, o2.Order_Type, COUNT(o2.Order_Type) AS Food_Count
FROM Order AS o1, Order AS o2
GROUP BY o1.Order_Type, o2.Order_Type
ORDER BY o1.Order_Type DESC, COUNT(o2.Order_Type) DESC;
# Data from query stored as object ‘df’

```r
> df_mexican <- subset(df, FoodType1 == "Mexican")
> df_mexican$percentage = paste(round(df_mexican$Count/sum(df_mexican$Count) *100), "%", sep=' ')
> pie(df_mexican$Count, labels = paste(df_mexican$FoodType2, df_mexican$percentage, sep=' '), main = "Distribution of Additional Orders Given One (1) Order of Mexican Food")
```
Query 4 - Promotional Effectiveness

Which promotion strategy is the most effective for a given location?

Procedure:
Compare the profit increase under different promotion schemes on different locations.

Utility:
✓ Suggestion system: can predict and suggest new restaurants to a user given his or her restaurant preferences
✓ Advertising optimization: advertisements for specific restaurants can be targeted to users who are more likely to purchase given their previous history
Query 4 - Promotional Effectiveness

SQL - Find the most effective promotion strategy for a given City

```sql
FROM Advertisement as a
GROUP BY a.City
FROM Advertisement as a1
WHERE a.City = a1.City));
```
Query 4 - Promotional Effectiveness

% conn to the database
conn = database('BerkeleyDeliversDB', '', ');
% Execute SQL statement and open cursor
curs = exec(conn, 'promotionSQL');
% Import data into MATLAB
curs = fetch(curs, 6);
% Get data
PE = curs.Data;
numOfRow = rows(curs);
promotion = java_array('java.lang.String', numOfRow);
city = java_array('java.lang.String', numOfRow);
profit = zeros(1, numOfRow);
xlabel = java_array('java.lang.String', numOfRow);
for i = 1:numOfRow
    promotion(i) = java.lang.String(PE(i, 1));
    city(i) = java.lang.String(PE(i, 2));
    profit(i) = PE(i, 3);
    s = strcat(promotion(i), ',', city(i));
    xlabel(i) = java.lang.String(s);
end
figure;
xlabel = cell(xlabel);
bar(profit, 'FaceColor',[0 .5 .5], 'EdgeColor',[0 .9 .9]);
set(gca, 'XTickLabel',xlabel, 'XTick',1:numol(xlabel));
Query 5 - Delivery Time Prediction

What is the estimated delivery time from a given business to a location on a given time slot?

Procedure:
Calculate the average delivery speed (miles per min) in different hour of the day from a given business to a ZIP Code, and then estimate delivery time.

Utility:
✓ Customer Service: Provide an estimate delivery time after customers placed order
✓ Performance metrics: Focus on the businesses that allow efficient delivery and butting out businesses that are inefficient
Query 5 - Delivery Time Prediction

SQL - Determine the average time to deliver an order to a particular ZIP Code at any given time

CREATE VIEW Total_Delivering_Time(Business_ID, ZIPCode, Ordering_Time, Total_Time, Delivering_Distance) AS
SELECT a.Business_ID, o.ZIPCode, o.Ordering_Time, (HOUR(o.Delivering_Time)*60+MINUTE(o.Delivering_Time) - HOUR(o.Ordering_Time)*60-MINUTE(o.Ordering_Time)) AS Total_Time, o.Delivering_Distance
FROM Business b, Order o, Product p

SELECT tt.Business_ID, tt.ZIPCode, HOUR(tt.Ordering_Time) AS OrderHour, AVG(tt.Total_Time) AS Average_Time
FROM Total_Delivering_Time AS tt
GROUP BY HOUR(tt.Ordering_Time), tt.Business_ID, tt.ZIPCode;
R code for **Delivery Time** in different time intervals: We randomly sampled 10 days’ delivery time data and overlapped them in one plot.

```r
pdf("~/Desktop/query5.pdf", width = 10, height = 6)
matplot(time_matrix, ave_matrix, type = "l", lty = 1, lwd = 2, ylab = "Delivery Time (min)", xlab = "Time Intervals", xaxt = "n", col = rgb(0,0.2,1,seq(0.2, 1, length=10)), main = "10 Samples Delivery Time For 5 Time Intervals")
abline(v = 5:10, col = "gray", lty = 2)
axis(1, at = (5:10), labels = c("5 pm","6 pm","7 pm","8 pm","10 pm","11 pm"))
mean = apply(ave_matrix, 1, function(x) mean(x))
lines(x = 5:10, y=mean, col = "red", lwd = 3, lty =2)
text(5.5,17, labels = "Average Delivery Time", col = "red", cex=1)
arrows(5.5,12.3, 5.5, 16.5, col = "red", cex = 0.8, lwd =2)
dev.off()
```

<table>
<thead>
<tr>
<th>Business_ID</th>
<th>ZIPCODE</th>
<th>Ordering_Tim</th>
<th>Delivering_Ti</th>
<th>Delivering_D</th>
<th>Total_Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>16324</td>
<td>94709</td>
<td>17:34</td>
<td>17:56</td>
<td>1.8</td>
<td>22</td>
</tr>
<tr>
<td>37206</td>
<td>94711</td>
<td>18:29</td>
<td>18:57</td>
<td>2.1</td>
<td>28</td>
</tr>
<tr>
<td>23051</td>
<td>94710</td>
<td>18:54</td>
<td>17:20</td>
<td>1.3</td>
<td>26</td>
</tr>
<tr>
<td>18395</td>
<td>94711</td>
<td>19:08</td>
<td>19:40</td>
<td>2.3</td>
<td>32</td>
</tr>
</tbody>
</table>
Average Delivery time (red dashed line) is plotted by using the coefficients generated by the delivery time for each time interval of operation. (5pm - 11pm)
R - Use data to predict delivery time given hour of day and distance to travel

#data stored as object ‘df2’
> df2$Hour <- as.factor(df2$Hour)
> lm_DeliveryTime <- lm(Delivery_Time ~ Hour + Distance, df2)
> coef(lm_DeliveryTime)

The regression equation predicted is:

Minutes to delivery = 6.1 + Distance*4.0 + If_5*0.1 + If_6*6.2 + If_7*12.8 + If_8*8.4 + If_9*8.2 + If_10*8.0 + If_11*7.9, where If_N is a set of binary variables indicating the hour.

Adj. $R^2 = 0.67$
Query 5 - Delivery Time Prediction

SQL - Implement Predicted Values

```
SELECT o.OID, Time_To_Delivery AS 6.1 + 4*tt.Delivering_Distance + IIF(HOUR(tt.Ordering_Time) = 5, 0.1, IIF(HOUR(tt.Ordering_Time) = 6, 6.2, IIF(HOUR(tt.Ordering_Time) = 7, 12.8, IIF(HOUR(tt.Ordering_Time) = 8, 8.4, IIF(HOUR(tt.Ordering_Time) = 9, 8.2, IIF(HOUR(tt.Ordering_Time) = 10, 8.0, IIF(HOUR(tt.Ordering_Time) = 11, 7.9, 0))))))))
FROM Order AS o, Product AS p, Total_Delivering_Time AS tt
```

![A table showing predicted delivery times](image)
Normalization
1NF Decomposition

Our table has no multi-valued attributes so we leave it as is:

Order (OID, Customer_ID\textsuperscript{1}, Order_Type, Placed_Time, SSN\textsuperscript{10}, Delivering_Time, Ordering_Time, Delivering_Distance, Employee_Num\textsuperscript{10}, Driver_license_Num\textsuperscript{10.2})
Decomposing from 1NF to 2NF:

Order (OID, Customer_ID, Order_Type, Placed_Time, SSN, Delivering_Time, Ordering_Time, Delivering_Distance, Employee_Num, Driver_license_Num)

Order (OID, Customer_ID, Order_Type, Placed_Time, Employee_Num, Delivering_Time, Ordering_Time, Delivering_Distance)

Employee_Handling(Employee_Num, Driver_license_Num, SSN)
Decomposing from 2NF to 3NF:

Order (OID, Customer_ID<sup>10</sup>, Order_Type, Placed_Time, Employee_Num<sup>10</sup>
Delivering_Time, Ordering_Time, Delivering_Distance)

Employee_Handling(Employee_Num<sup>10</sup>, Driver_license_Num<sup>10.2</sup>, SSN<sup>10</sup>)

All attributes in our tables are dependent on a primary key.
1NF Decomposition

Our table has no multi-valued attributes so we leave it as is:

Credit Card (Payment_No., Card_Num, Card_type, Name_on_card, Expire_date, Security_code, SSN, Employee_Num, Managed_Department)
Decomposing from 1NF to 2NF:

Credit Card (Payment_No.¹⁰, Card_Num, Card_type, Name_on_card, Expire_date, Security_code, SSN⁹, Employee_Num⁹, Managed_Department⁹.¹)

Credit Card (Payment_No.¹⁰, Card_Num, Card_type, Name_on_card, Expire_date, Security_code, Employee_Num²)

Processed_By (Employee_Num², SSN⁹, Managed_Department⁹.¹)
Decomposing from 2NF to 3NF:

Credit Card (Payment_No.¹⁰, Card_Num, Card_type, Name_on_card, Expire_date, Security_code, Employee_Num⁹)

Processed_By ( Employee_Num⁹, SSN⁹, Managed_Department⁹.¹ )

Credit Card (Payment_No.¹⁰, Card_Num, Employee_Num⁹)

Processed_By ( Employee_Num⁹, SSN⁹, Managed_Department⁹.¹ )

Card_Info ( Card_Num, Card_type, Name_on_card, Expire_date, Security_code )
Thank You!

Questions?