Tyler Heyl, Robby DiFabio, Evan Scott

Data-Mining Project Write-Up

Executive Summary

This report explores the data provided by Instacart in 2017 as part of a public competition to predict which products will be in a user's next order. Our analytical approach includes insights such as time-series comparisons of customer, product, and department traffic across working hours and days of the week, feature correlation, and relationship mapping for aspects such as popularity and association, to name a few. The goal is to extract meaningful insights, patterns, and conclusions by exploring customer purchasing behaviors from various facets. Utilizing ggplot2, SQL, and various other standard R libraries, we provide visualizations and analysis, breaking down the data and patterns from a descriptive analytics perspective. All technical work is in the R code uploaded to the group repository, while the data was left out due to file size constraints set by GitHub but can be downloaded here: https://www.kaggle.com/c/instacart/data.

Our Data

Working with the dataset provided by Instacart for their Instacart Market Basket Analysis competition, we started with a relational set of files describing three million grocery orders from more than 200,000 customers over time. These seven CSV files included 25 columns total and were initialized as follows:

- **Departments.csv**
- **Aisles.csv**
- **Order_products_*.csv (train & prior)**
- **Orders.csv**
- **Products.csv**
- **Sample_submission.csv**

From here, we dropped repetitive and/or unuseful files to our descriptive task. Specifically, we dropped 'sample_submission.csv' (file specific to the competition hosted by Instacart) and 'order_products_prior.csv' since it was a larger version of order_products_train.csv thus increasing the computational cost and heavily slowing execution time.

We identified and handled missing values and irrelevant columns to ensure the data was suitable for analysis. Of our remaining five files stored as data frames, missing values were only found in the orders table, specifically in the 'days_since_prior_order' column. Since it could not be deduced whether these missing values in this column represented first-time orders or simply information that was not provided, the result would have caused a reduction of just ~ 6% of the total records out of over 200,000, so we decided to omit them.

Next, we handled some column modifications. We dropped the 'eval_set' column as it was irrelevant to our analysis and created time-related columns 'order_day' and 'order_hour' from the ambiguous descriptions provided (0-6 for day and 0-23 for hour) to add context for understanding customer behavior across different periods.

We merged tables to reduce redundancies. Since the products table was the only one referencing the information provided by the aisles and departments tables, we merged them into products to reduce the number of tables to reference and the frequency of joins.
The resulting data we were to work with now includes 3 data frames with 18 columns total:

- **Orders_cleaned**
- **Products_cleaned**
- **Order_products_train**

Analysis

Visualization 1: Table

Some initial observations to take note of when observing the summary statistics of our data are listed in the table below. By tracking statistics on each order like the aisle and department food is kept, we can identify trends to enhance the predictive analysis abilities of the app. The average time between orders for all users was just over eleven days but ranged anywhere from later that same day to over 30 days later while the average order size was around 10 with a cart high of over 80 products. This wide spread of values in a number of the impactful variables causes complications and complexity when working with and attempting to gain insights out of their relationships thus requiring specific case handling to workaround.

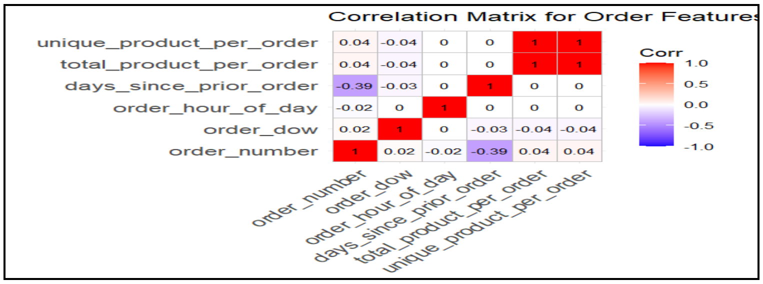
```
# Combine important metrics/aggregations from the summary statistics of the different tables into one structure
summary_table <- tibble(
  Metric = c('Orders', 'Days Between Orders (Avg)', 'Customers', 'Order Size (Avg)',
             'Products', 'Departments', 'Aisles'),
  Value = c(
    summary_orders$total_orders,
    round(summary_orders$avg_days_between_orders, 2),
    summary_orders$total_customers,
    round(avg_order_size, 2),
    summary_products$total_products,
    summary_products$total_departments,
    summary_products$total_aisles
  )
)

# Pretty print a table visualization
summary_table %>%
  kable() %>%
  kable_styling(bootstrap_options = 'striped', full_width = FALSE)
```

Metric	Value
Orders	3214874.00
Days Between Orders (Avg)	11.11
Customers	206209.00
Order Size (Avg)	10.55
Products	49688.00
Departments	21.00
Aisles	134.00

Visualization 2: Correlation Matrix

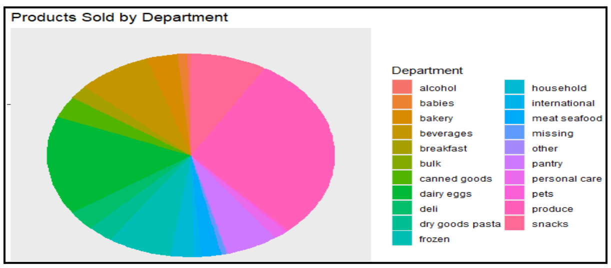
A correlation matrix is intended to visualize the strength and direction of relationships between variables of interest, specifically linear. However, as you can see with our visualization, the numeric values we extracted through the SQL query do not exhibit linear relationships. This can be explained by the nature of the variables we are working with. For example, order_dow encodes a categorical variable (day of the week), but this as a number simply does not provide meaning when drawing relationships. In other words, through this visualization, we were able to pick out this abnormality, the relationships are more complex and thus may benefit from other visualizations that can map and present these relationships, potentially nonlinear, better than a correlation matrix can.

```
corr_matrix <- cor(corr_numeric_data)

ggcorrplot(corr_matrix, lab = TRUE, lab_size = 3, title = 'Correlation Matrix for Order Features',
           colors = c('blue', 'white','red'))
```



Visualization 3: Pie Chart

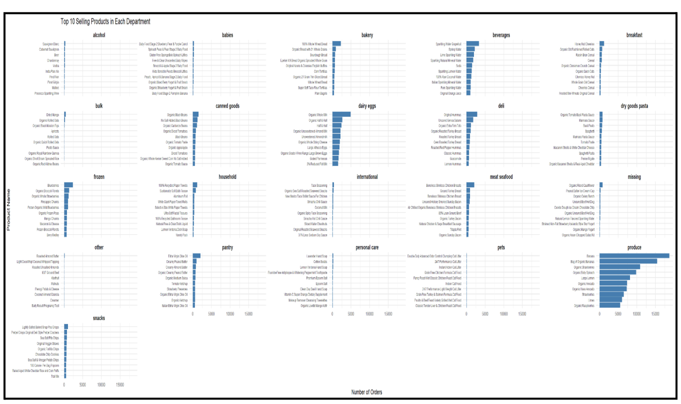
To get an accurate representation of the amount of products being sold in each grocery store department, we decided to make a pie chart. Grouping by department and calculating proportions with respect to other departments, we were able to view with this query and plot that the most popular departments are dairy and eggs, produce, and snacks. Those can be explained clearly by human behaviors. Dairy and eggs are essential for everyday cooking and baking, making them frequent purchases, while snacks are often bought on impulse. It looks like refrigerated goods are the most commonly purchased type of product on Instacart.




```r
ggplot(pie_data, aes(x = "", y = proportion, fill = department)) +
  geom_bar(stat = "identity") +
  coord_polar("y") +
  labs(
    title = "Products Sold by Department",
    x = NULL,
    y = NULL,
    fill = "Department"
  ) +
  theme(
    axis.text.x = element_blank(),
    axis.text.y = element_blank(),
    panel.grid = element_blank()
  )
```

Visualization 4: Histogram

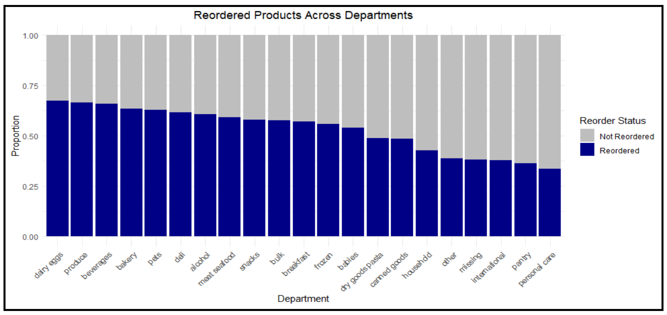
We created histograms providing us with the top ten most commonly purchased products for each department found in a grocery store. Using facet_wrap, we could look at each department's top-performing items and compare them to other departments. You can see that most departments have a favored good, which came as a surprise as we imagined that there would be more competition within departments rather than a stand-out product.



```r
#ggplot2 histogram visualization
top_items_by_department <- order_products_train %>%
  group_by(product_id) %>%
  summarise(total_orders = n()) %>%
  left_join(products_cleaned, by = "product_id") %>%
  group_by(department) %>%
  arrange(department, desc(total_orders)) %>%
  slice_head(n = 10) %>%  #top 10 products per department
  ungroup()
#facet-wrapped plot by department, showing most sold items by department
ggplot(top_items_by_department, aes(x = reorder(product_name, total_orders), y = total_orders)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  facet_wrap(~ department, scales = "free_y") +  #Allow independent y-axis scales
  labs(
    title = "Top 10 Selling Products in Each Department",
    x = "Product Name",
    y = "Number of Orders"
  ) +
  theme_minimal() +
  theme(
    axis.text.y = element_text(size = 6),   #Adjust product name text size
    axis.text.x = element_text(size = 8),   #Adjust x-axis tick label size
    strip.text = element_text(face = "bold", size = 10))  #Department labels styling
```

Visualization 5: Stacked Barchart

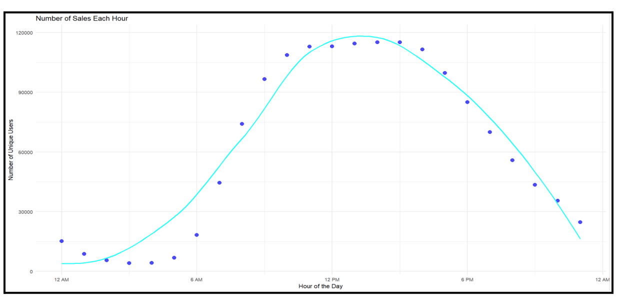
By reordering the contents of our SQL query we are able to draw a few relationships out of similar departments. There is a general decreasing trend in the proportion of reordered products as you move from left to right across the departments. Similar to what we mentioned previously with customer purchasing behaviors, customers are frequenting departments centralized on produce. While on the lower half, we see lower reordering rates which can be attributed to lower consumption/turnover like personal care items and miscellaneous things that. The usability of a product along with the rate at which it needs to be replenished appears to have a direct correlation with its reordering potential which is to be expected.




```r
ggplot(barchart_data, aes(x = reorder(department, -proportion * (reordered == 1)), y = proportion, fill = factor(reordered))) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("0" = "grey", "1" = "darkblue"), labels = c("Not Reordered", "Reordered")) +
  labs(
    title = "Reordered Products Across Departments",
    x = "Department",
    y = "Proportion",
    fill = "Reorder Status"
  ) +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1)
  )
```

Visualization 6: Scatterplot

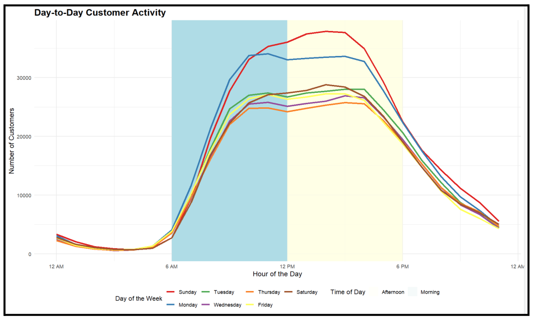
By leveraging ggplot(), geom_point(), geom_smooth(), and other operators, we were able to form a scatterplot that compares the number of orders at each hour of the day or in other words, the customer frequency. With geom_smooth(), we placed a smooth trend line over the plot to identify linear or non-linear relationships. Without a trend line, we would also be unable to see any patterns not shown by the variability in the data. Most orders coming late morning into the afternoon make sense, as people may be shopping for items they will cook for dinner that night.




```r
#ggplot2 Scatterplot visualization
#Finding amount of sales per hour each day
hourly_orders <- order_data %>%
  group_by(order_hour_of_day, order_dow) %>%
  summarise(unique_users = n_distinct(user_id)) %>%

ggplot(hourly_orders, aes(x = order_hour_of_day, y = unique_users)) +
  geom_point(color = "blue", size = 3, alpha = 0.7) +
  geom_smooth(method = "loess", color = "cyan", se = FALSE) +  #Smoothed trend line
  scale_x_continuous(
    breaks = c(0, 6, 12, 18, 24),
    labels = c("12 AM", "6 AM", "12 PM", "6 PM", "12 AM")
  ) +
  labs(
    title = "Number of Sales Each Hour",
    x = "Hour of the Day",
    y = "Number of Unique Users"
  ) +
  theme_minimal()
```

Visualization 7: Line Chart

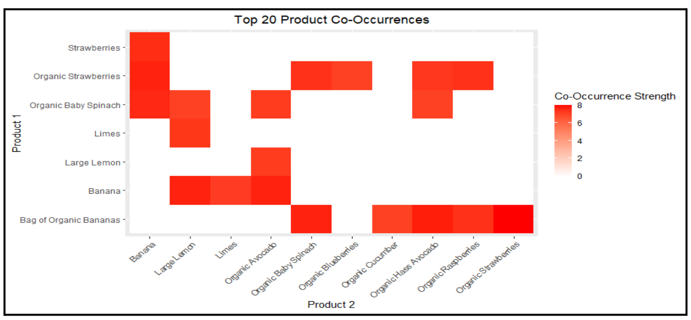
By using ggplot(), geom_rect(), and geom_line(), we created a line chart to compare the amount of periodic grocery store activity on each day of the week. Using geom_rect(), we could distinguish even further the peak times, which are the morning and afternoon. We also distinguished each day of the week by color and used scale_color_brewer() to label the numeric variable with the true days of the week. It was expected that Sunday would be the most popular day of the week to grocery shop due to most people not working, and this assumption was correct. What was shocking, however, was the similarities between Monday and Sunday's activity. Monday's activity proves to be much larger than every day except Sunday.




```r
#Finding amount of customers per hour each day
hourly_customers <- orders_cleaned %>%
  group_by(order_dow, order_hour_of_day) %>%
  summarise(unique_users = n_distinct(user_id))


#Line Chart creation
ggplot(hourly_customers, aes(x = order_hour_of_day, y = unique_users, color = factor(order_dow))) +
  #Add transparent shaded areas
  geom_rect(aes(xmin = 6, xmax = 12, ymin = -Inf, ymax = Inf, fill = "Morning"), alpha = 0.1, inherit.aes = FALSE) +  #Morning
  geom_rect(aes(xmin = 12, xmax = 18, ymin = -Inf, ymax = Inf, fill = "Afternoon"), alpha = 0.1, inherit.aes = FALSE) +  #Afternoon
  geom_line(size = 1.2) +  # Customer activity lines
  scale_fill_manual(
    name = "Time of Day",
    values = c("Morning" = "lightblue", "Afternoon" = "lightyellow")
  ) +
  scale_x_continuous(
    breaks = c(0, 6, 12, 18, 24),
    labels = c("12 AM", "6 AM", "12 PM", "6 PM", "12 AM")
  ) +
  labs(
    title = "Day-to-Day Customer Activity",
    x = "Hour of the Day",
    y = "Number of Customers",
    color = "Day of the Week"
  ) +
  scale_color_brewer(palette = "Set1", labels = c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")) +
  theme_minimal() +
  theme(
    legend.position = "bottom",
    plot.title = element_text(size = 16, face = "bold"),
    plot.subtitle = element_text(size = 12),
    axis.title = element_text(size = 12))
```

Visualization 8: Heatmap

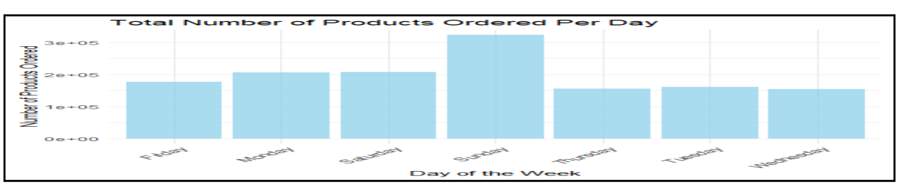
The heatmap highlights patterns in customer purchasing behavior, helping to identify which products are often bought together. It helps track customer habits and preferences like common meal combinations. Think of fruit for example, organic strawberries and regular strawberries are often purchased together, showing one of the strongest co-occurrences. This suggests customers balance preferences for organic options with price or availability. It is also offers consideration for arranging items and aisles in such a way that products complement each other and drive positive trends in sales because of customer interest and attention.




```r
# Aggregate products that occur in together in an order
co_occurrence <- sqldf('
  SELECT p1.product_id AS product_id_1,
         p2.product_id AS product_id_2,
         COUNT(*) AS co_occurrence
  FROM order_products_train p1
  JOIN order_products_train p2 ON p1.order_id = p2.order_id AND p1.product_id < p2.product_id
  GROUP BY p1.product_id, p2.product_id
')

# Filter for top 20 co-occurrences
top_co_occurrence <- co_occurrence[order(-co_occurrence$co_occurrence), ][1:20, ]

# Join to get the product names for plot visualization interpretability
top_co_occurrence <- sqldf('
  SELECT c.*, p1.product_name AS product_name_1, p2.product_name AS product_name_2
  FROM top_co_occurrence c
  JOIN products_cleaned p1 ON c.product_id_1 = p1.product_id
  JOIN products_cleaned p2 ON c.product_id_2 = p2.product_id
')
```

Visualization 9: Bar Chart

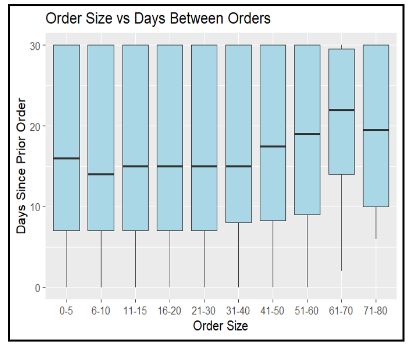
Is there potential for providing deals and recommendations on specific days to help further drive sales? These insights can help Instacart predict a user's next purchase by leveraging specific shopping patterns tied to the days of the week. For Tuesdays and Wednesdays, when many users are busy and non-active, Instacart can suggest smaller purchases, like snacks or quick meal ingredients, and offer reminders or midweek discounts as encouragement. There are obvious advantages of knowing the high-traffic days like the weekend.



Visualization 10: Boxplot

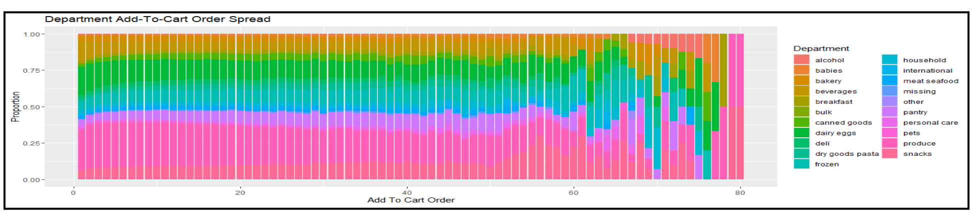
Instacart can predict that users who place smaller orders will shop more frequently, often within 1-2 weeks, and can suggest replenishment items. For users who place larger orders, Instacart can expect less frequent purchases, about every 3-4 weeks, and recommend bulk or pantry staples. The visual highlights the median, showing that larger orders generally have longer intervals between purchases.

```r
# Order the buckets for ggplot
boxplot_data$order_size_in_buckets <- factor(
  boxplot_data$order_size_in_buckets,
  levels = c('0-5', '6-10', '11-15', '16-20', '21-30',
             '31-40', '41-50', '51-60', '61-70', '71-80', '80+'),
  ordered = TRUE
)

head(boxplot_data)

ggplot(boxplot_data, aes(x = factor(order_size_in_buckets), y = days_since_prior_order)) +
  geom_boxplot(fill = 'lightblue') +
  labs(title = 'Order Size vs Days Between Orders',
       x = 'Order Size',
       y = 'Days Since Prior Order')
```



Visualization 11: Stacked Histogram

This stacked histogram illustrates the proportion of items added to the cart across various grocery store departments as order size increases with each color representing a specific department. Larger orders (to the right) include a broader mix of items like snacks (pink), pantry (red), and beverages (blue), indicating more diverse shopping habits. Lower cart order counts tend to follow similar trends in terms of item positions with departments while higher counts tend to be more random and sporadic possibly due to the intent of purchase (for an event, you might get many random items that aren;t normally coupled but still required) and range of products selected.



Visualization 12: Treemap

To create the Treemap visualization, we first needed to install the Treemapify package. Then, we could left-join order_producst_train to products_clean to analyze the sales per department in stores. This plot did not bring many surprises, as it was expected that produce and dairy/eggs would be frontrunners in sales as those items are extremely popular and abundant in United States grocery stores. We were surprised at how small of a share of the plot that personal care got however, because this is an essential department that almost every person uses. This could be because people may turn to alternate stores, such as enterprises which specialize in personal care items, rather than Instacart.


```r
#ggplot2 Treemap creation
library(treemapify)

department_sales <- products_cleaned %>% #New table for plotting purposes
  left_join(order_products_train, by = "product_id") %>%
  count(department)
ggplot(department_sales, aes(area = n, fill = department, label = department)) +
  geom_treemap() +
  geom_treemap_text(fontface = "bold", color = "white", place = "centre", grow = TRUE) +
  labs(title = "Department Sales") +
  theme_minimal()
```

Queries

Query 1: Summary statistics of the cleaned dataset

We ran queries to analyze the patterns in our datasets once they were cleaned. In this query, we summarized all of the variables in the dataset.

```r
summary_orders <- sqldf('
                        SELECT COUNT(*) AS total_orders,
                        AVG(days_since_prior_order) AS avg_days_between_orders,
                        COUNT(DISTINCT user_id) AS total_customers
                        FROM orders_cleaned'
                        )

avg_order_size <- sqldf('
                        SELECT AVG(order_size) AS avg_order_size
                        FROM (
                          SELECT order_id, COUNT(*) AS order_size
                          FROM order_products_train
                          GROUP BY order_id
                          )'
                        )

avg_order_size_val <- avg_order_size$avg_order_size

summary_products <- sqldf('SELECT COUNT(*) AS total_products,
                          COUNT(DISTINCT department) AS total_departments,
                          COUNT(DISTINCT aisle) AS total_aisles
                          FROM products_cleaned'
                          )
```

```
summary_products
total_products total_departments total_aisles
         49688                21          134
summary_orders
total_orders avg_days_between_orders total_customers
     3214874                11.11484          206209
avg_order_size
avg_order_size
      10.55276
```

Query 2: Correlation coefficients between numeric variables of interest

For this query, we extract all relevant numeric variables to be put into a correlation matrix in order to assess the coefficients for potential linear relationships along with the strength and direction of them if applicable.

```r
customer_order_sum <- sqldf('
                            SELECT order_id,
                            COUNT(*) AS total_product_per_order,
                            COUNT(DISTINCT product_id) AS unique_product_per_order
                            FROM order_products_train
                            GROUP BY order_id'
                            )

# Combine numeric columns from orders_cleaned and order_products_train to visualize
# any linear relationships via correlation
corr_numeric_data <- sqldf('
                           SELECT o.order_number, o.order_dow, o.order_hour_of_day,
                           o.days_since_prior_order, s.total_product_per_order, s.unique_product_per_order
                           FROM orders_cleaned o
                           INNER JOIN customer_order_sum s ON o.order_id = s.order_id'
                           )
```

```
> head(corr_numeric_data)
  order_number order_dow order_hour_of_day days_since_prior_order total_product_per_order unique_product_per_order
1           11         4                 8                     14                      11                       11
2           15         1                11                     30                      31                       31
3            5         0                11                      6                       9                        9
4           21         2                11                      6                       9                        9
5            4         1                14                     10                      18                       18
6            4         6                10                     30                      22                       22
```

Query 3: Comparison of department popularity

In this query, we want to look at the proportion of department popularity in percentage terms and products sold to better grasp department share. Our findings make sense as produce and dairy/eggs are large departments with many customers. It is surprising how little of a percentage of orders that the smaller departments get, such as pets and bulk.

```
> pie_data
   department_id      department products_sold   proportion
1              4         produce        409087 0.2954513775
2             16      dairy eggs        217051 0.1567588727
3             19          snacks        118862 0.0858446776
4              7       beverages        114046 0.0823664595
5              1          frozen        100426 0.0725298043
6             13          pantry         81242 0.0586747093
7              3          bakery         48394 0.0349511814
8             15    canned goods         46799 0.0337992383
9             20            deli         44291 0.0319879071
10             9 dry goods pasta         38713 0.0279593563
11            17       household         35986 0.0259898586
12            12    meat seafood         30307 0.0218883634
13            14       breakfast         29500 0.0213055307
14            11   personal care         21570 0.0155783152
15            18          babies         14941 0.0107907096
16             6   international         11902 0.0085958789
17            21         missing          8251 0.0059590486
18             5         alcohol          5598 0.0040429953
19             8            pets          4497 0.0032478295
20             2           other          1795 0.0012963874
21            10            bulk          1359 0.0009814989
```

```r
pie_data <- sqldf('
                  SELECT p.department_id, p.department, COUNT(o.product_id) AS products_sold
                  FROM order_products_train o
                  JOIN products_cleaned p ON o.product_id = p.product_id
                  GROUP BY p.department_id
                  ORDER BY products_sold DESC'
                  )

# Calculate department popularity proportion against all departments
pie_data$proportion <- pie_data$products_sold / sum(pie_data$products_sold)

pie_data
```

Query 4: Top Performing product in each Department

For the fourth query, we wanted to get a snapshot of what the top products were in each department. Finding out exactly what item is being purchased the most posed many insights, as it was shocking that Sparkling Water Grapefruit was the most popular beverage, as we thought it would be a soda or water.

```
# SQL query to find the top product per department (Histogram)
top_item_per_department <- "SELECT department, product_name
FROM (
  SELECT
  p.department,
  p.product_name,
  COUNT(o.product_id) AS total_orders,
  ROW_NUMBER() OVER (PARTITION BY p.department ORDER BY COUNT(o.product_id) DESC) AS rank
  FROM
  order_products_train o
  JOIN
  products_cleaned p
  ON
  o.product_id = p.product_id
  GROUP BY
  p.department, p.product_name
) AS subquery
WHERE rank = 1"
sqldf(top_item_per_department)
```

	department	product_name
1	alcohol	Sauvignon Blanc
2	babies	Baby Food Stage 2 Blueberry Pear & Purple Carrot
3	bakery	100% Whole Wheat Bread
4	beverages	Sparkling Water Grapefruit
5	breakfast	Honey Nut Cheerios
6	bulk	Dried Mango
7	canned goods	Organic Black Beans
8	dairy eggs	Organic Whole Milk
9	deli	Original Hummus
10	dry goods pasta	Organic Tomato Basil Pasta Sauce
11	frozen	Blueberries
12	household	100% Recycled Paper Towels
13	international	Taco Seasoning
14	meat seafood	Boneless Skinless Chicken Breasts
15	missing	Organic Riced Cauliflower
16	other	Roasted Almond Butter
17	pantry	Extra Virgin Olive Oil
18	personal care	Lavender Hand Soap
19	pets	Double Duty Advanced Odor Control Clumping Cat Litter
20	produce	Banana
21	snacks	Lightly Salted Baked Snap Pea Crisps

Query 5: Likelihood of reordering based on department

In this query, we look at how often departments can expect to get their products reordered versus not reordered. We can see larger disparities for departments where there is high consumption and turnover of its products such as with baking goods and alcohol as there is a much larger reorder rate associated. This is also attributed to the items being ordered having a higher product count in general.

```
# Aggregate by department and reorder status to calculate proportion
barchart_data <- sqldf('
  SELECT p.department,
         o.reordered,
         COUNT(*) AS product_count,
         COUNT(*) * 1.0 / SUM(COUNT(*)) OVER (PARTITION BY p.department) AS proportion
  FROM order_products_train o
  JOIN products_cleaned p ON o.product_id = p.product_id
  GROUP BY p.department, o.reordered
')

head(barchart_data)
```

```
> head(barchart_data)
```

	department	reordered	product_count	proportion
1	alcohol	0	2201	0.3931761
2	alcohol	1	3397	0.6068239
3	babies	0	6857	0.4589385
4	babies	1	8084	0.5410615
5	bakery	0	17702	0.3657891
6	bakery	1	30692	0.6342109

Query 6: Customers by the Hour

This query shows us the hours of the day people order food the most. The findings were not shocking, as it makes sense that the late morning and early afternoon have the most orders.

```
#sqldf query to see how many sales by the hour

hourly_sales <- "SELECT
    order_hour_of_day,
    COUNT(DISTINCT user_id) AS unique_users
  FROM
    orders_cleaned
  GROUP BY
    order_hour_of_day
  ORDER BY
    order_hour_of_day"
sqldf(hourly_sales)
```

```
> sqldf(hourly_sales)
```

	order_hour_of_day	unique_users
1	0	15281
2	1	8854
3	2	5549
4	3	4225
5	4	4280
6	5	6866
7	6	18352
8	7	44579
9	8	74119
10	9	96666
11	10	108667
12	11	112986
13	12	113025
14	13	114484
15	14	115060
16	15	115105
17	16	111551
18	17	99676
19	18	85006
20	19	70025
21	20	55804
22	21	43537
23	22	35528
24	23	24778

Query 7: Peak Hours of Each Day of the Week

This query shows us the top two hours of every day of the week, along with their respective customer amount. What was slightly surprising is that Monday, having the second most customers per day, has different peak hours than every other day of the week.

```
# Query to find the top 2 hours(for sake of output length) with the most unique customers for each day of the week
top_hours_by_day <- "SELECT day_of_week, order_hour_of_day, unique_customers
  FROM (
    SELECT
      CASE order_dow
        WHEN 0 THEN 'Sunday'
        WHEN 1 THEN 'Monday'
        WHEN 2 THEN 'Tuesday'
        WHEN 3 THEN 'Wednesday'
        WHEN 4 THEN 'Thursday'
        WHEN 5 THEN 'Friday'
        WHEN 6 THEN 'Saturday'
      END AS day_of_week,
      order_hour_of_day,
      COUNT(DISTINCT user_id) AS unique_customers,
      ROW_NUMBER() OVER (PARTITION BY order_dow ORDER BY COUNT(DISTINCT user_id) DESC) AS rank
    FROM
      orders_cleaned
    GROUP BY
      order_dow, order_hour_of_day
  ) AS ranked_hours
  WHERE rank <= 2"
sqldf(top_hours_by_day)
```

```
> sqldf(top_hours_by_day)
   day_of_week order_hour_of_day unique_customers
1       Sunday                14            37888
2       Sunday                15            37708
3       Monday                11            34082
4       Monday                10            33767
5      Tuesday                16            28037
6      Tuesday                15            28006
7    Wednesday                15            26913
8    Wednesday                16            26565
9     Thursday                15            25744
10    Thursday                16            25541
11      Friday                14            27269
12      Friday                15            27267
13    Saturday                14            28778
14    Saturday                15            28426
>
```

Query 8: Items frequently bought together

This query displays common pairings of products or, in other words, which items are frequently bought together. Ranking at the top of this list is the pairings of bananas with the likes of strawberries, avocado, and spinach. This can serve as an indicator of healthy choices where a customer's purchasing pattern can be predicted via nutritional value and produce association.

```
# Aggregate products that occur in together in an order
co_occurrence <- sqldf('
  SELECT p1.product_id AS product_id_1,
         p2.product_id AS product_id_2,
         COUNT(*) AS co_occurrence
  FROM order_products_train p1
  JOIN order_products_train p2 ON p1.order_id = p2.order_id AND p1.product_id < p2.product_id
  GROUP BY p1.product_id, p2.product_id
')

# Filter for top 20 co-occurrences
top_co_occurrence <- co_occurrence[order(-co_occurrence$co_occurrence), ][1:20, ]

# Join to get the product names for plot visualization interpretability
top_co_occurrence <- sqldf('
  SELECT c.*, p1.product_name AS product_name_1, p2.product_name AS product_name_2
  FROM top_co_occurrence c
  JOIN products_cleaned p1 ON c.product_id_1 = p1.product_id
  JOIN products_cleaned p2 ON c.product_id_2 = p2.product_id
')
```

```
> top_co_occurrence
   product_id_1 product_id_2 co_occurrence         product_name_1         product_name_2
1         13176        21137          3074 Bag of Organic Bananas Organic Strawberries
2         13176        47209          2420 Bag of Organic Bananas Organic Hass Avocado
3         13176        21903          2236 Bag of Organic Bananas Organic Baby Spinach
4         24852        47766          2216                 Banana      Organic Avocado
5         21137        24852          2174   Organic Strawberries               Banana
6         24852        47626          2158                 Banana          Large Lemon
7         21903        24852          2000   Organic Baby Spinach               Banana
8         16797        24852          1948           Strawberries               Banana
9         13176        27966          1780 Bag of Organic Bananas  Organic Raspberries
10        21137        27966          1670   Organic Strawberries  Organic Raspberries
11        21137        21903          1639   Organic Strawberries Organic Baby Spinach
12        26209        47626          1595                  Limes          Large Lemon
13        21137        47209          1539   Organic Strawberries Organic Hass Avocado
14        21903        47766          1402   Organic Baby Spinach      Organic Avocado
15        47626        47766          1349            Large Lemon      Organic Avocado
16        24852        26209          1331                 Banana                Limes
17        21137        39275          1269   Organic Strawberries  Organic Blueberries
18        13176        30391          1268 Bag of Organic Bananas      Organic Cucumber
19        21903        47209          1252   Organic Baby Spinach Organic Hass Avocado
20        21903        47626          1238   Organic Baby Spinach          Large Lemon
```

Query 9: Product popularity by day of the week

In this query, we are looking at the number of products purchased on each separate day of the week. Looking at our findings, it is not surprising that the end of the week has fewer purchases, as most customer may be completing their grocery shopping at the beginning of the week.

```
bar_data <- sqldf('
  SELECT o.order_day, COUNT(p.product_id) AS num_products
  FROM orders_cleaned o
  JOIN order_products_train p ON o.order_id = p.order_id
  GROUP BY o.order_day
')
```

```
> bar_data
  order_day num_products
1    Friday       176910
2    Monday       205978
3  Saturday       207279
4    Sunday       324026
5  Thursday       155481
6   Tuesday       160562
7 Wednesday       154381
```

Query 10: Order size versus days since the prior order

For this query, we look at the amount of items in a single order in comparison to the days since the user last ordered. Through this, we can analyze if there is a correlation between the size of the order and the frequency of their order history. The findings do not translate to much correlation, which may be because people make specific orders based on personal decisions and ways in which they cook, varying in amount and people whom they are buying for.

```
boxplot_data <- sqldf("
                    SELECT o.user_id, o.order_number, o.days_since_prior_order,
                    CASE
                      WHEN s.order_size BETWEEN 0 AND 5 THEN '0-5'
                      WHEN s.order_size BETWEEN 6 AND 10 THEN '6-10'
                      WHEN s.order_size BETWEEN 11 AND 15 THEN '11-15'
                      WHEN s.order_size BETWEEN 16 AND 20 THEN '16-20'
                      WHEN s.order_size BETWEEN 21 AND 30 THEN '21-30'
                      WHEN s.order_size BETWEEN 31 AND 40 THEN '31-40'
                      WHEN s.order_size BETWEEN 41 AND 50 THEN '41-50'
                      WHEN s.order_size BETWEEN 51 AND 60 THEN '51-60'
                      WHEN s.order_size BETWEEN 61 AND 70 THEN '61-70'
                      WHEN s.order_size BETWEEN 71 AND 80 THEN '71-80'
                      ELSE '80+'
                    END AS order_size_in_buckets
                    FROM orders_cleaned o
                    JOIN (
                      SELECT order_id, COUNT(product_id) AS order_size
                      FROM order_products_train
                      GROUP BY order_id
                    ) s ON o.order_id = s.order_id"
                    )
```

```
> head(boxplot_data)
  user_id order_number days_since_prior_order order_size_in_buckets
1       1           11                     14                 11-15
2       2           15                     30                 31-40
3       5            5                      6                  6-10
4       7           21                      6                  6-10
5       8            4                     10                 16-20
6       9            4                     30                 21-30
```

Query 11: Positions that products get added to the cart at

This query joins the necessary tables then calculates the proportion that departments commonly get added to the cart at a specific position. As one example, this can be interpreted that beverages are added to a customer's cart first roughly 14% of the time.

```
> head(stacked_hist_data)
  add_to_cart_order department count  proportion
1                 1    alcohol  1268 0.009663971
2                 1     babies   857 0.006531564
3                 1     bakery  4625 0.035249106
4                 1  beverages 18073 0.137742076
5                 1  breakfast  2144 0.016340343
6                 1       bulk   138 0.001051757
```

```
combined_df <- sqldf('
              SELECT o.add_to_cart_order, p.department
              FROM order_products_train o
              LEFT JOIN products_cleaned p ON o.product_id = p.product_id')

# Calculate the proportion departments are added to a customer's cart at specific spots
stacked_hist_data <- sqldf('
              SELECT add_to_cart_order, department, COUNT(*) AS count,
              COUNT(*) * 1.0 / SUM(COUNT(*)) OVER (PARTITION BY add_to_cart_order) as proportion
              FROM combined_df
              GROUP BY add_to_cart_order, department
              ORDER BY add_to_cart_order, department'
              )
```

Query 12: Total purchases in Each Department

In this SQL query, we look at department popularity based on the number of orders from each department. The findings were not shocking, as the stand-out departments are produce and dairy/eggs, which is what we predicted.

```
# create new table to compare sales in each department
department_sales <- products_cleaned %>%
  left_join(order_products_train, by = "product_id") %>%
  count(department)
sales_by_department <- "SELECT department, n
      FROM department_sales
      ORDER BY n DESC"
sqldf(sales_by_department)
```

```
> sqldf(sales_by_department)
        department      n
1          produce 409238
2       dairy eggs 217467
3           snacks 120106
4        beverages 114885
5           frozen 101019
6           pantry  82506
7           bakery  48597
8     canned goods  47176
9             deli  44494
10 dry goods pasta  39045
11       household  36744
12    meat seafood  30437
13       breakfast  29683
14   personal care  23819
15          babies  15153
16   international  12158
17         missing   8604
18         alcohol   6005
19            pets   4714
20           other   1970
21            bulk   1362
>
```

Conclusion

By using data analysis techniques, we explored aspects of customer behavior related to the Instacart experience. We learned working with data of this scale can quickly accelerate to large computational costs and slow execution times, but insights derived from this analysis are highly actionable. Understanding customer patterns and product trends which are made available via querying and visualizations can directly drive sales with marketing strategies and inventory management. We saw potential continuation of this exploration with the incorporation of aspects like predictive modeling which then could provide personalized recommendations to customers, ultimately leading to improved business outcome and revenue.