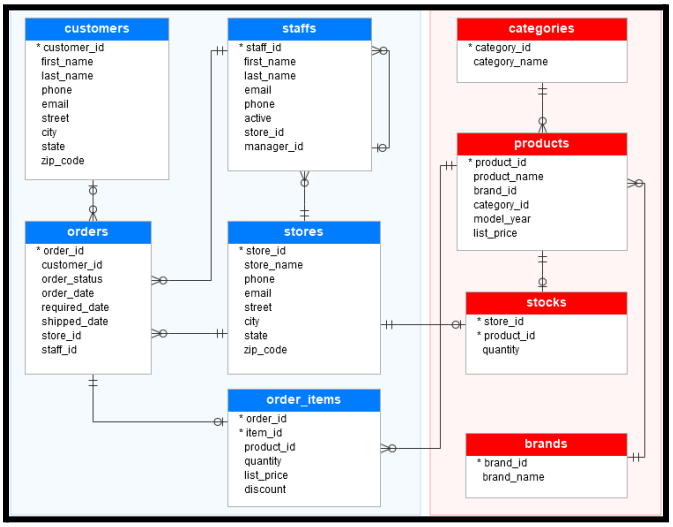
Brianna Floyd
Anastasia Lomtadze
Deric Woo
Dr. Levkoff
BUAN 370
15 December 2024

Data Mining Project Written Report

1. Executive Summary

To begin, we will provide a walk through of our data collection process, describe the variables of interest, and discuss the overall structure of our dataset. We collected our data for this project from kaggle, a website for data scientists that provides many example datasets to practice on. Our dataset is about a bike store chain, listing sales data for orders and specific items, as well as data on staff, stores, products, categories, and inventory. We used every table except for stocks (inventory) and brands, for a total of 7 tables. In the image below is an Entity Relationship Diagram showing the relationships between each of the tables.



To summarize, the main table we considered was orders. In each order, there can be zero or more order items. In each order there can only be one of each item, because here they are using item_id to count the number of items in each order, so “this bike is item 1 of order 1”. There can be 0 or 1 customer for each order, two customers cannot order together. There can be zero or more orders per customer, meaning some customers can return later to make another purchase. Only one staff member can work on a particular order, but multiple orders might be completed by one staff member. A staff member can only work at one store, but a store can have zero or more staff members. Each order item has a field called product_id that links to the products table. That relationship is one to one (because it’s essentially to itself), but the same product can exist in multiple order item lines. The products table links to the categories table. Each product belongs to one category, but each category will have multiple products.

In this project, we utilized SQL queries, visualizations in R, and linear models to explore and analyze this data. The sections that follow will detail the specific steps of our process.

2. Data Cleaning

We did not have to deal with erroneous missing values in our dataset. We checked all the fields for nulls and found none. As a second check, we decided to look for any observations listed as “NULL” as a string. We did find some, only in the shipped date column. When we investigated

further, we found that these “NULL”s were only present in records where the order status was less than 4, meaning the order had not yet shipped. It makes perfect sense that we wouldn’t have a listed shipping date if the order has not yet shipped, so we didn’t need to fix this.

We did not have to rename any column headers or recode data at the beginning, however, later on in the project, we created columns for year and month of order date, which required naming. We did not have to transform variables, but we did use the function as.factor() on several over the course of the project.

Below are the two queries mentioned that we used to check for NULLs and “NULL”s.

```
SELECT COUNT(*) FROM orders_df WHERE shipped_date = 'NULL'
```

```
SELECT COUNT(*) FROM orders_df WHERE shipped_date = 'NULL' AND order_status = 4
```

Next, we looked into what sort of data we had in each table and thought about how we could use them.

The “categories.csv” dataset consists of two variables: “category_id” primary key and “category_name”. The first is numeric, sequential, and uniquely identifies each category, with a uniform distribution. The second consists of categorical values representing different types of bikes. This dataset has no duplicates or missing values, and is evenly distributed.

The “customers.csv” dataset contains several unique identifiers such as primary key “customer_id”, and “email”, name variables like “first_name” and “last_name”, and address components, like “street”, “city”, “state”, and “zip code”. The “phone” variable is mostly empty, which makes it less useful compared to other variables for comprehensive analysis.

The dataset “order_items.csv” consists of the “order_id” primary key, which is a numeric identifier for each order, and “item_id” primary key, a sequential, numeric identifier of each item within an order. “product_id” is a numeric id for what the customer is actually buying, while “quantity” is a numeric variable showing the number of units of the product. The “list_price” is a numeric variable indicating the price per unit, ranging widely. “discount” is a numeric variable representing the discount percent applied to an item, written as a decimal.

The “orders.csv” dataset is related to customer orders and provides insight into the delivery status of the order along with customer information. We consider this our main table. The “order_id” primary key uniquely identifies the transaction of each order, while “customer_id” links the order to the corresponding customer. The progress or the completion of the order is tracked by the “order_status” variable, and the “order_date”, “required_date”, and “shipped_date” track the timeline for each process. “store_id” indicates what store the order came from, and the “staff_id” identifies the employee responsible for the sale.

The “products.csv” dataset contains information about the types of bike products, each having a unique sequential id and primary key “product_id”, and a corresponding unique “product_name”. The “brand_id” is a numeric id that connects to a manufacturer, “category_id” categorizes the bikes by type, “model_year” indicates year of production, and “list_price” is the product’s retail price.

The dataset “staffs.csv” provides details about staff members, each uniquely identified by the primary key “staff_id” variable. There are “first_name”, “last_name”, and “email” fields listing employee information. The “phone” variable lists unique phone numbers with area codes reflecting the location of the store they work for. The “active” variable is binary, indicating whether an employee is currently working or not. The “store_id” shows which store an employee works at. Lastly, the “manager_id” connects the employees to their respective managers.

The “stores.csv” dataset includes unique identifiers like the primary key “store_id” and “store_name” variables. This data helps understand which variables are related to which store. The “phone” and “email” columns provide contact details for each location. Additionally, there are

columns like “street”, “city”, “state”, and “zip_code”, that provide details about the address of each store.

We investigated what data types we were working with. The code in the image below shows the results for the orders dataframe.

```
class(orders_df$order_id) # numeric
class(orders_df$customer_id) # numeric
class(orders_df$order_status) # numeric
class(orders_df$order_date) # Date
class(orders_df$required_date) # Date
class(orders_df$shipped_date) # character (because some observations are 'NULL')
class(orders_df$store_id) # numeric
class(orders_df$staff_id) # numeric
```

We learned that the shipped date field was not actually in Date format. This is because there were some observations of the string “NULL” from earlier. We considered changing those observations to be actually NULL, and therefore able to make this column in Date format, but we decided it only made sense to do this if it became necessary later.

3. Exploratory Data Analysis

This third section will detail our exploratory analysis on the cleaned data. We utilized SQL queries and the ggplot2 package to develop 10 visualizations that provide a visualized narrative regarding what is going on in our bike store data. Several of our queries are associated with the visualizations we made. We made sure to create many different types of visualizations to fully understand and present the data in an engaging way.

We did look into relationships between variables through queries and by creating visualizations, but we also used linear modeling. We created 3 linear models. First, for the effect that the staff id had on order price, so does which staff member explain higher or lower sales. Second, for store id on order price, so does which store explain higher or lower sales. Third, for the month’s effect on sales, so does which month it is explain a difference in average sales. This was the only model that resulted in statistically significant results. In the image below are the results. We can see three stars for several months, which means the sales numbers are significantly different (at the 5% level) in these months compared to January (shown here as the intercept). The estimate for each variable is negative, so this means that January usually has better sales than all the other months. We wonder if this could be attributed partially to people making New Year's resolutions to exercise more.

```
Residuals:
    Min      1Q  Median      3Q     Max
-4157.9 -2130.7  -675.7  1572.0 12891.1

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)       4408.9      223.7  19.709  < 2e-16 ***
monthFebruary    -1022.9      326.2  -3.136 0.001747 **
monthMarch       -1003.2      300.5  -3.339 0.000863 ***
monthApril       -1434.7      358.4  -4.003 6.57e-05 ***
monthMay         -1157.2      350.0  -3.307 0.000968 ***
monthJune         -514.7      350.0  -1.471 0.141607
monthJuly        -1327.4      355.1  -3.738 0.000193 ***
monthAugust      -1363.5      336.8  -4.049 5.43e-05 ***
monthSeptember    -860.1      337.6  -2.548 0.010941 *
monthOctober     -1410.9      330.8  -4.265 2.13e-05 ***
monthNovember     -688.5      360.7  -1.909 0.056504 .
monthDecember    -1138.2      355.1  -3.205 0.001379 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2758 on 1433 degrees of freedom
  (170 observations deleted due to missingness)
Multiple R-squared:  0.02313,   Adjusted R-squared:  0.01563
F-statistic: 3.084 on 11 and 1433 DF,  p-value: 0.0004148
```

Now, we will showcase the main queries that we wrote to explore our datasets. We ran 17 queries in total in our code to demonstrate our ability to use SQL and the sqldf package. Below is the syntax for those queries, minus the two included in Section 2.

```
SELECT order_date, COUNT(order_id) AS order_count
FROM orders_df
GROUP BY order_date
ORDER BY COUNT(order_id) DESC
LIMIT 5
```

The query above aimed to answer the question: Why was the count of orders in one month in our first visualization so high? We thought maybe there was one day with a huge amount of orders, perhaps even by mistake. The highest count of orders, however, was 9, which is reasonable. And a fun fact, April 19th is National Biking Day, which could be an explanation for a high order count in April. Another factor: spring is also a popular time for biking because of the nice weather.

```
SELECT DISTINCT(staff_id)
FROM orders_df
```

In **Query 4**, we wanted to see how many total staff members there are by using the DISTINCT function in SQL.

```
SELECT staff_id, COUNT(order_id) AS order_count
FROM orders_df
GROUP BY staff_id
ORDER BY COUNT(order_id) DESC
LIMIT 3
```

In **Query 5**, we used SQL to see which staff member is the most successful in selling bikes. It is similar to our **Query 14**, where we looked at which staff member brought in the most sales. This query still shows the same outcome, where the top two sales staff are six and seven. Number six, Marcelene Boyer, has sold 553 bikes and Venita Daniel has sold 540 bikes.

```
SELECT first_name, last_name, COUNT(order_id) AS order_count
FROM orders_df
LEFT JOIN staffs_df
USING (staff_id)
GROUP BY staff_id
ORDER BY COUNT(order_id) DESC
LIMIT 3
```

In **Query 6**, it is the same as the last query, but we joined the orders dataframe with the staffs dataframe to get the names of the staff members in the same dataframe.

```
SELECT category_name, COUNT(*)
FROM order_items_df
LEFT JOIN products_df USING (product_id)
LEFT JOIN categories_df USING (category_id)
GROUP BY category_id
ORDER BY COUNT(*) DESC
LIMIT 10
```

In **Query 7**, we explored to see what the most popular category of the bikes that were sold are. Through this code, we were able to conclude that Cruisers Bicycles were the most popular category of bicycles. This seems reasonable as cruiser bicycles are more of a casual, common and user-friendly bicycle that everybody from all walks of life can use and enjoy.

```
query_8 <- "SELECT product_id, order_date
FROM order_items_df
LEFT JOIN orders_df USING (order_id)
WHERE product_id IN (
SELECT product_id
FROM order_items_df
GROUP BY product_id
ORDER BY COUNT(*) DESC
LIMIT 5)
ORDER BY order_date"
query_8 <- sqldf(query_8)
query_8 <- as.data.frame(query_8)
query_8 <- query_8 %>%
  mutate(year = year(order_date), month = month(order_date)) %>%
  group_by(product_id, year, month) %>%
  summarise(order_count = n(), .groups = 'drop')
```

In **Query 8**, we were trying to figure out the most popular bikes over time. It is also one of our visualizations, where you can see that throughout the couple years, there are bigger order spikes in 2016 versus a big drop off in 2017.

```
query_9 <- "SELECT order_id, order_date, store_id, SUM(list_price) AS order_price
FROM order_items_df
LEFT JOIN orders_df USING (order_id)
GROUP BY order_id"
query_9 <- sqldf(query_9)
query_9 <- as.data.frame(query_9)
query_9 <- query_9 %>%
  mutate(year = year(order_date), month = month(order_date)) %>%
  group_by(store_id, year, month) %>%
  summarise(order_price = n(), .groups = 'drop')
```

In **Query 9**, our code depicts the sales over time. Similar to the bar graph for **Query 11**, we see that store ID two is higher than one and three, which is showing that New York sales have always been higher than California and Texas. There is a spike in 2018 earlier in the year and starts to drop off towards the end of it.

```
SELECT DISTINCT customer_id, COUNT(customer_id) AS customer_id_count
FROM orders_df
GROUP BY customer_id
ORDER BY customer_id_count DESC
LIMIT 39
```

In **Query 10**, the query above, we are trying to find out which customers buy the most from the bike company; they could also be described as the most loyal customers. There are 39 customers who have ordered a maximum amount of orders of three. We ran this query to see what the maximum order count is and how many customers keep returning to purchase bikes from the company.

```
query_11 <- full_join(order_items_df, orders_df, by = 'order_id') %>%
  group_by(order_date) %>%
  mutate(product_count = sum(quantity)) %>%
  mutate(month = month(order_date), year = year(order_date))
```

In **Query 11**, this is the code that we used to determine the count of bikes sold over time by store. We were curious to see how the location and time of the store impacted the count of sales. New York was the most popular, then it was California, then Texas. For graphs 1 and 2, as the year went on, there were fewer bikes purchased, but graph 3 didn't seem to follow the same trend.

```
query_12 <- full_join(order_items_df, orders_df, by = 'order_id')

query_12 %>%
  select(product_id, list_price, quantity) %>%
  group_by(product_id) %>%
  mutate(total = quantity * list_price) %>%
  summarize(total_rev = sum(total)) %>%
  arrange(desc(total_rev))
```

In **Query 12**, we were exploring to see what product brought in the most money. Through this code, we were able to determine that product number 7, also known as the "Trek Slash 8 27.5 - 2016" has made the most money with $615,998.

```
query_13 <- full_join(order_items_df, orders_df, by = 'order_id')

query_13 %>%
  select(list_price, quantity) %>%
  mutate(total = quantity * list_price) %>%
  summarize(sum(total))
```

In **Query 13**, the goal was to figure out how much total revenue that the bikes were bringing in. The total amount of money being made was $8,578,989.

```
query_14 <- full_join(order_items_df, orders_df, by = 'order_id')

query_14 %>%
  select(list_price, quantity, staff_id) %>%
  group_by(staff_id) %>%
  mutate(total = quantity * list_price) %>%
  summarize(total_rev = sum(total)) %>%
  arrange(desc(total_rev))
```

In **Query 14**, we solved how much revenue each employee brought in and who brought in the most. Through our code, we were able to conclude that employee number 6, Marcelene Boyer brought in the most money with $2,938,889. Venita Daniel was a close second, bringing in $2,887,353.

```
query_15  <- "WITH ranked_products AS (
  SELECT store_name, product_name, COUNT(*) AS product_count,
         ROW_NUMBER() OVER (PARTITION BY store_name ORDER BY COUNT(*) DESC) AS rank
  FROM order_items_df
  LEFT JOIN orders_df USING (order_id)
  LEFT JOIN stores_df USING (store_id)
  LEFT JOIN products_df USING (product_id)
  GROUP BY store_name, product_name
)
SELECT store_name, product_name, product_count
FROM ranked_products
WHERE rank = 1
ORDER BY store_name;"
```

In **Query 15**, we wanted to see which bikes were more or less popular at different stores. In this query, we had to join orders_items to orders, stores, and products because we needed information at the item level for products and stores. We saw that the *Electra Cruiser 1 (24-Inch) - 2016* was the most popular at Baldwin Bikes and Rowlett Bikes, but not at the third store.

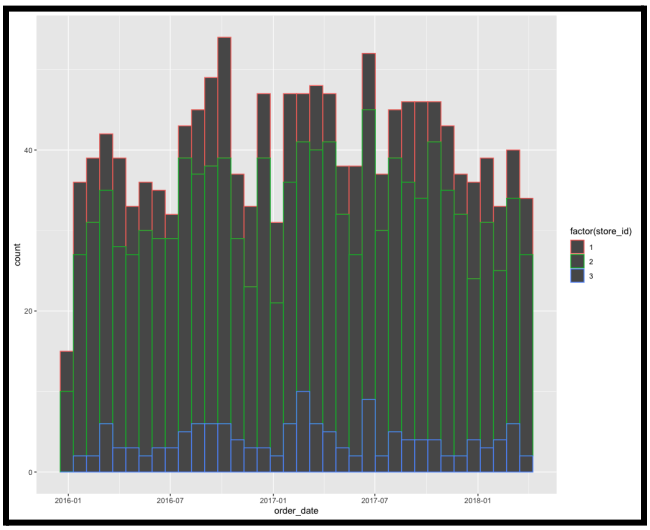
```
SELECT product_id, products_df.list_price, category_id, category_name
FROM products_df
LEFT JOIN categories_df USING (category_id)
```

Query 16 was used to generate a data frame for Visualization 10, the box plot.

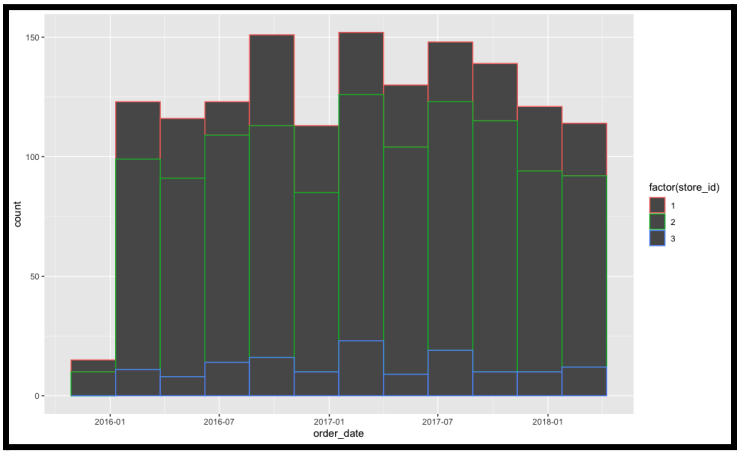
```
SELECT order_id, order_date, store_id, staff_id, SUM(list_price) AS order_price
FROM order_items_df
LEFT JOIN orders_df USING (order_id)
GROUP BY order_id
```

Query 17 was used to generate a table with all the data we needed for our linear models.

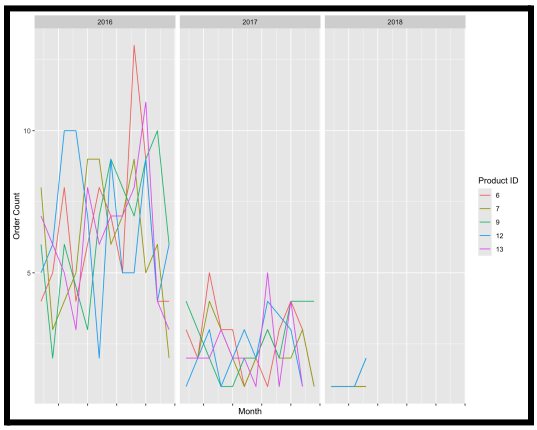
Next, below are the visualizations that we made for this project. We used facet wrappers and colors to distinguish between stores, categories, staff members, and more variables.



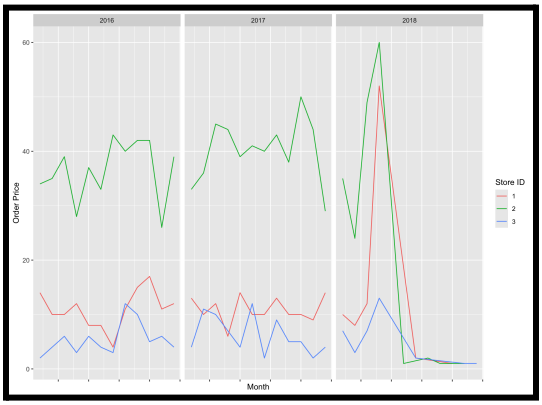
The above graph provides histograms measuring the number of orders received colored by each store id. We used 36 bins, for the months in three years.



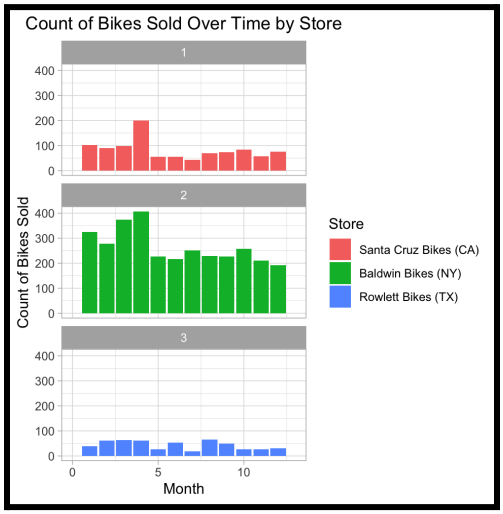
The above graph is the same as the previous one, only this time, we used 12 bins, for the quarters in three years. It appeared visually more normal this way, which leads us to believe the data is not so inconsistent.



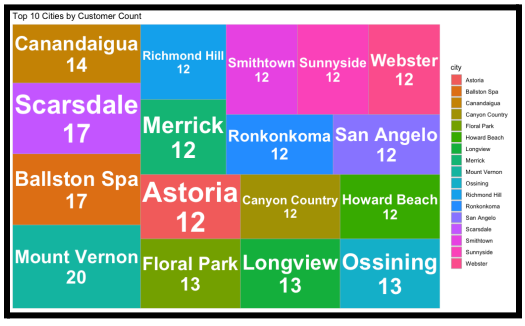
The visualization above is based off of **Query 8** and it depicts the number of orders received over time by product only for the top 5 products (top meaning most orders received).



The visualization above is based on **Query 9** and shows the average order price over time. Color differentiates the store id.

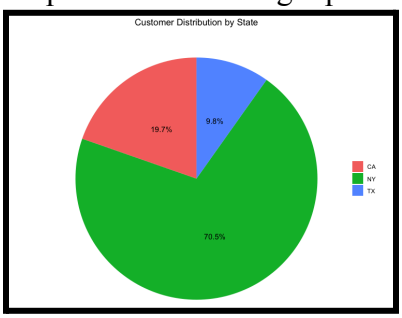


Taking a look at the graph above, which is generated from **Query 11**, we're taking a look at the count of bikes sold over time by each store. There are only three stores to take a look at: one in California, one in New York, and one in Texas. In graphs one and two, there seems to be a bit of a positive skew, showing that bikes are more popular in the earlier months in the year. For graph number three, the count of bikes sold is much lower in Texas in comparison to the other two. New York is significantly higher than both of the other states, showing much more popularity than the two in bikes sold. They have a high of a little over 400, whereas California has a high of roughly 200, and Texas has a high of somewhere merely over 50.



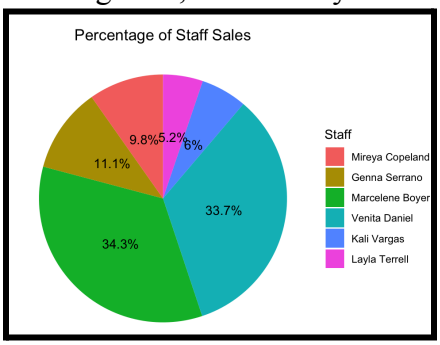
This visualization above is a treemap which displays the top 10 cities with the highest customer counts. The data is grouped by the *city* column, and the total number of customers in each city is calculated using the *summarize()* function. The cities are then sorted in descending order based on their customer counts.

The treemap, created using *ggplot2*, shows rectangles with different sizes corresponding to the number of customers in a city. The larger rectangles indicate higher customer counts, with each rectangle being filled in distinct colors and labeled. For instance, Mount Vernon counts the highest number of customers, highlighting its prominence among top cities.

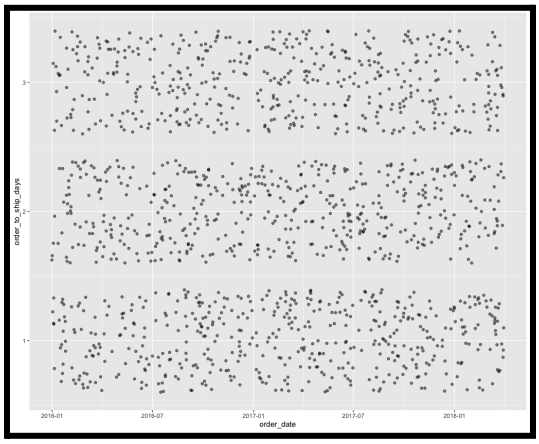


This graph above visualizes the distribution of customers by state using a pie chart. After grouping the data by *state* and calculating the total number of customers for each state, the date is then sorted in descending order to identify states with the largest customer density.

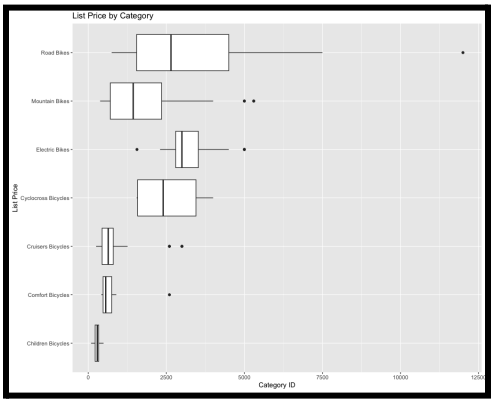
Each slice of the pie represents a state, with the size proportional to its customer count. The resulting visualization provides a clear representation of how customers are distributed across states, emphasizing New York as the dominating state, followed by California, and lastly, Texas.



This visualization above builds on **Query 14** to highlight the highest performing staff members. The percentages are in terms of contribution to total revenue.



The above graph showcases the distribution of the amount of time between an order being placed and the order being shipped. We used jitter and an alpha argument to make it so more of the points could be visible on the graph. The data points are clustered around the 1, 2, and 3 day areas, because we measure these dates in days, not hours or smaller. It's good to know that no order took longer than 3 days after the order date to ship.



For the graph above, we decided it would be best to show the distribution of sales per staff member using a pie chart. We used the dplyr and ggplot2 packages to first get the percentages of the amount of sales per staff member. To get the percentages, we used the total amount of revenue, divided by the sum of the total revenue. The total revenue was estimated by using the listing price times the count of the specific product that was sold, since there was no specific sales variable. Taking a look at **Figure 12**, we can see that the green takes up the majority of the pie chart, who is employee number 6; also known as Marcelene Boyer. The next employee would be Venita Daniel, coming in a close second. Surprisingly, there are only six out of a total of ten employees who are actually making sales.

4. Conclusion

In this project, we successfully utilized data mining techniques to analyze a Kaggle dataset about a bike store chain. Through our data cleaning process, we ensured the TIDYness and accuracy of our data, which allowed us to perform meaningful exploratory data analysis afterward. Our visualizations provided valuable insights into sales trends over time, store productivity, and staff performance. We identified key patterns, such as the distribution of orders over time, the most loyal customers, and the top-performing staff members. One significant finding was the higher popularity of bikes in New York compared to California and Texas. Additionally, our analysis highlighted the efficiency of the order processing and shipping system, with most orders being shipped within 3 days. Overall, this project demonstrated the power of data mining and data analysis in uncovering trends and outliers, which can help make informed decisions. Lessons we learned include the importance of thorough data cleaning (more complex than simply deleting all rows with null values), the value of different types of data visualizations like pie charts and tree maps, and the need for organization of code with useful, readable comments and uniform variable naming conventions.