# ADMIN FINAL REPORT


## Rudi's Bakery Route Optimization: An Integrated Approach using Machine Learning and Optimization Modeling

**Master of Science in Business Analytics**

**Course: Experiential Projects**

**S24 – 004 – Group 4 – Project 1**

**Prof. Noah Zikmund**

## Group Members

**Amandeep Bhardwaj**
(amandeep.bhardwaj@colorado.edu)

**Anubhav Dubey**
(anubhav.dubey@colorado.edu)

**Sai Arvind Atluri**
(saiarvind.atluri@colorado.edu)

**Sandeep Reddy Modugu**
(sandeep.modugu@colorado.edu)

## Table of Contents

## Business Understanding

Rudi's Bakery, renowned for its high-quality gluten-free and organic bread products, is at a pivotal juncture in enhancing its logistical operations in the Colorado Springs area. The company's Direct Store Delivery (DSD) system plays a crucial role in its supply chain, facilitating direct deliveries to stores to ensure product freshness and availability. As the business expands and market dynamics evolve, Rudi's is faced with the challenging task of optimizing these routes. This optimization goes beyond merely reducing costs; it aims to establish a sustainable model that balances driver workload, meets the delivery expectations of retail partners, and supports the company's strategic growth objectives. By improving operational efficiency, Rudi's can lessen its environmental impact through reduced fuel consumption and enhance employee satisfaction by ensuring fair work distribution.

## Problem Statement

The central challenge lies in devising an efficient and adaptable routing system capable of responding to fluctuating store demands while keeping logistical costs in check. The existing routing system, albeit functional, does not fully exploit the potential to increase store visits - a factor directly linked to sales growth. The objective is to reassess and redesign the DSD routes to boost the frequency of store visits without significantly increasing operational expenses. This requires an exhaustive analysis of store locations, delivery windows, driver schedules, and vehicle capacities, all within the constraints imposed by real-world logistics, such as traffic patterns and delivery timings.

## Overarching Objectives

**a) Maximizing Sales through Increased Store Visits**

A primary aim of optimizing the DSD routes is to enhance Rudi's Bakery's operational ability to increase the frequency of its store visits. By modifying the routes to enable each store to be visited three to five times per week—up from the current minimum of three—the company anticipates improvements in product visibility and availability. This, in turn, is expected to boost customer satisfaction and potentially increase sales by 5%. This strategy acknowledges the direct correlation between the frequency of store visits and sales volume, using logistical efficiency as a lever for business growth.

**b) Minimizing Operational Costs**

Simultaneously, the project seeks to refine operations to conserve time and resources on deliveries. This involves reducing the distance travelled and optimizing driver schedules to cut down on fuel consumption and labor costs. By creating more efficient routes, Rudi's stands to decrease operational expenses, thereby contributing to the company's overall profitability. Achieving this goal demands a sophisticated route planning approach that incorporates advanced analytics and route optimization modeling to find the most cost-effective routes that still meet service level expectations.

## Data Overview

- **Data Collection:** The process begins with the systematic gathering of delivery route data from the client. This step ensures a comprehensive dataset that encompasses all aspects of the delivery operations, including historical delivery records, sales performance data, and a registry of store addresses.

- **Data Cleaning**: Once collected, the data undergoes meticulous cleaning to identify and rectify any errors or omissions. This phase involves removing duplicate entries, correcting typographical errors in addresses, and supplementing missing information. These actions are vital for the accuracy of subsequent analyses.

- **Data Validation**: Each store's address is authenticated using the Google Maps Geocoding API. This fundamental step is undertaken to uncover discrepancies in the dataset, such as outdated addresses or inaccuracies in data entry.

- **Geocoding Transformation**: This process converts textual addresses into precise geographic coordinates (latitude and longitude). It is facilitated by a custom-developed Python function that interfaces with the Google Maps Geocoding API, setting the stage for precise distance and time calculations.

- **Spatial Analysis Preparation:** With explicit geographic coordinates available, the next step is to prepare for spatial analysis. This involves organizing the coordinates in a structured manner to facilitate efficient processing in subsequent stages, such as the generation of distance and time matrices.

- **Distance Matrix Generation**: Utilizing the geographic coordinates, a distance matrix is generated via the Google Maps Directions API. This matrix calculates the physical distances between each pair of delivery locations, taking into account possible routes and road networks.

- **Time Matrix Generation:** We developed a time matrix integrating real-time data to estimate travel times between delivery locations, ensuring its accuracy across the entire matrix within a margin of variance no greater than 5%, thereby facilitating robust and efficient route planning.

- **Data Integration:** This phase involves the seamless incorporation of live data from the Google Maps API into the previously established matrices. The dynamic nature of this data allows for adjustments based on current traffic patterns and road conditions, adding a responsive component to the optimization process.

- **Optimization Matrix Preparation:** With the distance and time matrices in place, the data is now primed for optimization. This preparation stage organizes the matrices in a manner compatible with optimization algorithms, ensuring the data can be effectively processed to identify optimal delivery routes.

# Model Formulation

## Methodology

➢ **K-Means Clustering Algorithm**

**Purpose**: To group delivery points into clusters based on similarity in travel time. This step simplifies the route optimization by reducing the number of variables and constraints that the model needs to consider.

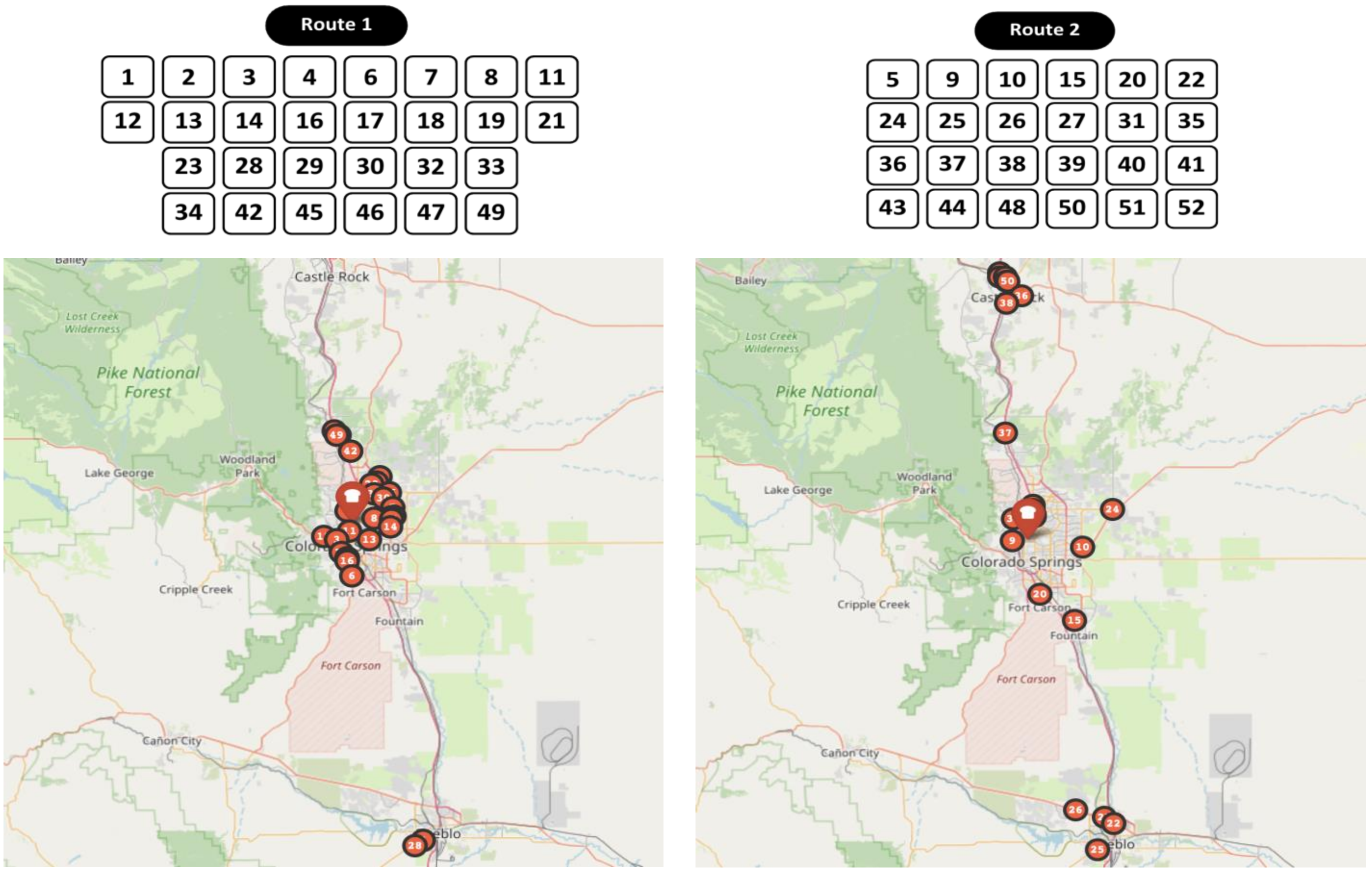
➢ **Gurobi Optimizer**

**Purpose**: To determine the most efficient delivery route that minimizes total travel time while satisfying a set of logistical constraints and decision variables.

## What was done?

➢ **Strategic Clustering with K-Means Clustering Algorithm**

After preparing the time data, the next step was to cluster the delivery points into distinct groups. The K-Means clustering algorithm was employed to organize stores into clusters, ensuring that retailers within the same cluster had relatively shorter average travel times among them compared to those in different cluster. This strategy was adopted to simplify the route optimization process that followed. By dividing the delivery points into clusters, the optimization framework could focus on developing efficient routes within each smaller, more manageable group of stores. Besides aiming to reduce computational complexity, this clustering also sought to create logically organized routes, potentially resulting in shorter travel times.



- **Route Optimization with Gurobi Optimizer**

For route optimization, we used Gurobi, a powerful tool for linear and integer optimization problems, with the delivery points divided into clusters. This approach effectively planned delivery routes by building upon the cluster foundation. It aimed to minimize the total travel time while considering the practical constraints surrounding the delivery operations. These constraints included ensuring each store was visited exactly once, adhering to the number of working hours within legal limits, and scheduling deliveries on specific days as required by some stores. The complexity of this model allowed for a balance between practicality and operational efficiency, ensuring that the planned routes were timely and in accordance with business policies and drivers' functional capabilities.

```
# Initialize the model
model = Model("VehicleRouteOptimization")

# Decision Variables
# Paths taken between stores on given days
visit_var = model.addVars(
    working_days, [(i, j) for i, j in time_matrix.keys() if i != j],
    vtype=GRB.BINARY, name="visit"
)

# Deliveries made to stores on given days
delivery_var = model.addVars(
    working_days, store_details['Store_ID'].unique(),
    vtype=GRB.BINARY, name="delivery"
)

### The Objective: Minimize Travel Time
model.setObjective(
    quicksum(visit_var[day, i, j] * time_matrix[(i, j)]
             for day in working_days for i, j in time_matrix.keys() if i != j),
    GRB.MINIMIZE
)
```

We introduced two sets of binary decision variables: one set to indicate whether a store received a delivery on a specific day, and another to represent the existence of a delivery route between stores on any given day. Our objective was to minimize the overall travel time for all routes, ensuring that each business was served within the operational days without revisiting any store on the same day. Through this approach, the model identified the most efficient set of routes to minimize total travel time while adhering to the determined constraints, thereby enhancing service levels and streamlining delivery operations.

## Constraints

In optimizing delivery routes, several constraints were applied to ensure the solution is viable and meets operational requirements:

- **Visit Constraints:** Each store, excluding the depot, must have exactly one entry and one exit in the route plan. This ensures that every store is visited precisely once. The model achieves this by applying constraints that require the sum of binary decision variables—indicating the use of a route between two points—to equal 1 for each store's incoming and outgoing paths.

```
# Assuming depot is store 0
for day in working_days:
    # Ensure the depot is left first
    model.addConstr(quicksum(visit_var[day, 0, j] for j in stores if j != 0) == 1, f"StartFromDepot_{day}")
    # Ensure the depot is entered last
    model.addConstr(quicksum(visit_var[day, i, 0] for i in stores if i != 0) == 1, f"ReturnToDepotLast_{day}")

# Single exit and entry for each store, ensuring path continuity
for day in working_days:
    for store in stores[1:]:  # Excluding the depot
        model.addConstr(quicksum(visit_var[day, i, store] for i in stores if i != store) >= delivery_var[day, store], f"MinIncoming_{day}_{store}")
        model.addConstr(quicksum(visit_var[day, store, j] for j in stores if j != store) >= delivery_var[day, store], f"MinOutgoing_{day}_{store}")
```

- **Delivery Days Constraint**: Certain stores, such as Costco, have specific delivery days due to contractual or functional requirements. The model addresses this by setting the binary delivery choice variables for these stores to 1 on all required delivery days, ensuring daily deliveries are made. For the rest of the stores, it ensures at least 3 deliveries per week.

```
# Delivery requirements for Costco and other stores
for store in stores[1:]:
    if store in costco_stores:
        for day in working_days:
            model.addConstr(delivery_var[day, store] == 1, f"CostcoDelivery_{day}_{store}")
    else:
        model.addConstr(quicksum(delivery_var[day, store] for day in working_days) >= 3, f"MinDeliveries_{store}")
```

- **Working Hours Constraint**: The model incorporates a constraint to ensure that the total travel time (multiplied by the decision variables) plus fixed loading times for each day's route do not exceed the maximum working hours. This prevents the assignment of more deliveries than can be realistically completed within a day, considering both loading and travel times.

```
# Working hours constraint, adjusted for minimized total travel time
for day in working_days:
    model.addConstr(
        quicksum((time_matrix[(i, j)] + loading_time) * visit_var[day, i, j] for i in stores for j in stores if i != j) <= working_hours_per_day,
        f"WorkingHours_{day}"
    )
```

- **Consecutive Delivery Constraint:** To avoid placing excessive demands on any store or route, the model prohibits deliveries to any store for three consecutive days. This is managed through a set of constraints ensuring a store cannot receive deliveries on the third day if it has already been delivered to on the first two consecutive days.

```
# Prevent delivery after two consecutive days with a more flexible approach
for store in stores:
    for i in range(len(working_days) - 2):
        day1, day2, day3 = working_days[i], working_days[i + 1], working_days[i + 2]
        model.addConstr((delivery_var[day1, store] + delivery_var[day2, store] + (1 - delivery_var[day3, store])) >= 1,
                        f"FlexibleNoConsecutiveDelivery_{store}_{day1}_{day2}_{day3}")
```

- **Load Balance Constraints:** The model strives to distribute the delivery workload evenly across available working days by setting a daily delivery limit. This is crucial for efficient resource allocation and to prevent drivers from being overburdened on any particular day.

```
# Constraint to limit the number of deliveries per day to less than 19
for day in working_days:
    model.addConstr(
        quicksum(delivery_var[day, store] for store in stores) <=18,
        f"MaxDeliveriesPerDay_{day}"
    )
```

## Costs using the technologies

- **Gurobi License Fee:** Acquiring a Gurobi license is a significant cost, especially for commercial operations. The cost varies based on the type of license and the scale of usage.

- **Google Maps API Charges:** While there's a free tier, extensive use of Google Maps API for generating real-time travel distances and times can lead to charges. The costs depend on the number of API calls made.

- **Operational Costs:** Beyond software licenses, operational costs such as vehicle maintenance, fuel, and driver salaries must be considered. Efficient route planning can help minimize these costs by reducing travel distance and time.

## Implementation of the Model

Gurobi's implementation of the model utilized a sophisticated methodology, benefiting from an enhanced dataset acquired through rigorous data management phases, including collection, cleaning, validation, and geocoding. The first step in this process was to employ the K-Means method for clustering, which organized delivery points into meaningful groups based on similar travel times. This initial step significantly reduced the complexity encountered during optimization by focusing on manageable clusters, thus improving the efficiency of route planning.

Following the clustering, Gurobi embarked on the optimization phase, aiming to minimize the total travel time within a framework of logistical constraints. These constraints—ensuring only one visit per store, adhering to set working hours, and compliance with delivery schedules—were meticulously integrated into the model to reflect real-world conditions. The optimization model used binary decision variables to represent the selection or exclusion of routes and deliveries, systematically navigating through these constraints.

In analysing and identifying the most efficient routing paths that met all predefined criteria, Gurobi applied both linear and integer programming techniques. This rigorous analysis confirmed that the devised supply routes aligned with both operational logistics and strategic business goals while being optimized for time efficiency.

The application's outcome revealed optimal routes that not only promised to reduce travel times but also to enhance delivery efficiency. This demonstrates the effectiveness of combining comprehensive data preparation with Gurobi's wholesome logistical optimization capabilities.

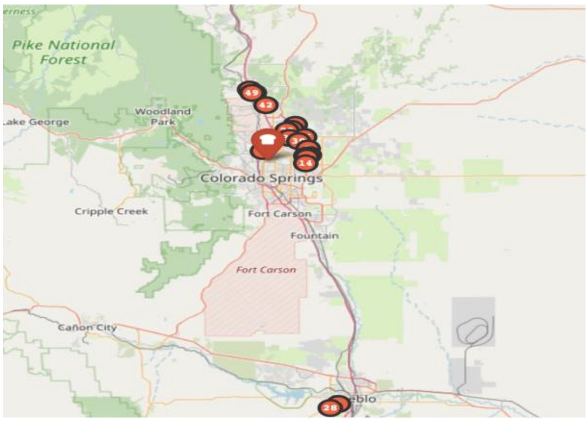
# Final Suggestion

## ➢ Suggested Route 1

| Delivery Points | Deliveries Per Week | Weekly Mileage | Weekly Duration |
|---|---|---|---|
| 28<br>Delivery Points | 88<br>Deliveries | 755.91<br>Miles | 44.14<br>Hours |

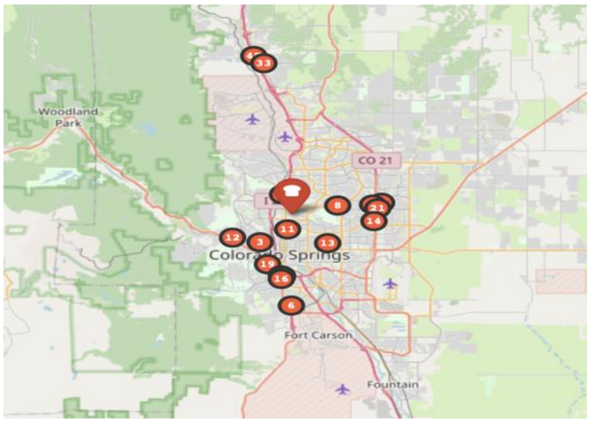
**Monday**

0 → 1 → 2 → 21 → 4 → 28 → 7
→ 17 → 14 → 29 → 32 → 30 → 46 →
33 → 45 → 34 → 47 → 42 → 49 → 0



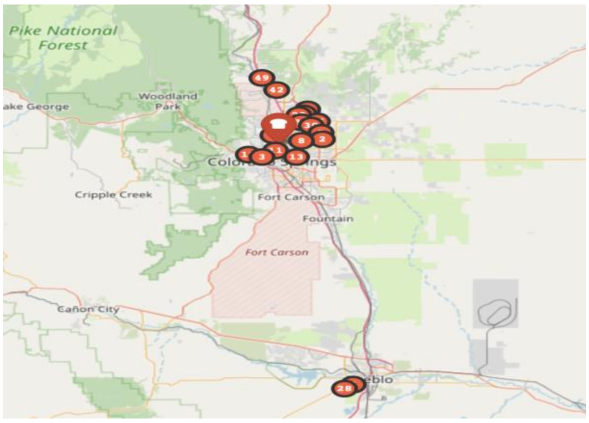

**Tuesday**

0 → 1 → 8 → 2 → 21 → 3 → 12
→ 6 → 16 → 13 → 11 → 14 → 17 →
19 → 18 → 23 → 33 → 45 → 0



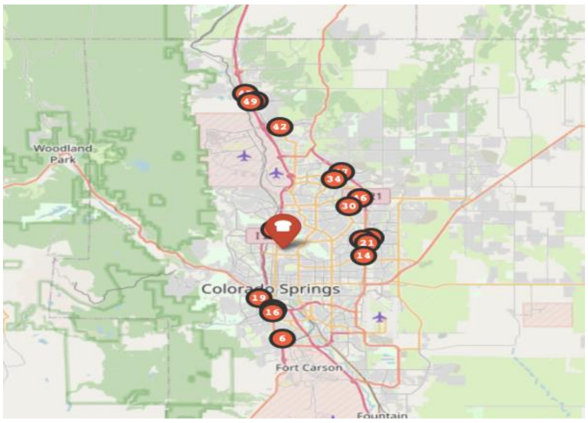

**Thursday**

0 → 1 → 8 → 2 → 7 → 3 → 12
→ 4 → 28 → 13 → 11 → 29 → 32 →
30 → 46 → 34 → 47 → 42 → 49 → 0



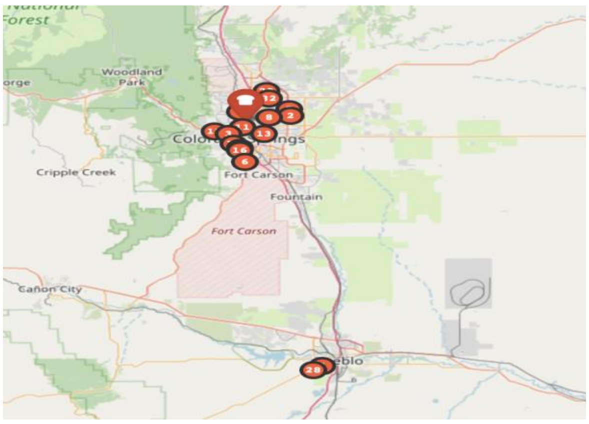

**Friday**

0 → 1 → 2 → 21 → 6 → 16 → 14
→ 17 → 19 → 18 → 23 → 30 → 46 →
33 → 45 → 34 → 47 → 42 → 49 → 0




**Saturday**

0 → 1 → 8 → 2 → 7 → 3 → 12
→ 4 → 28 → 6 → 16 → 13 → 11 →
19 → 18 → 23 → 29 → 32 → 0

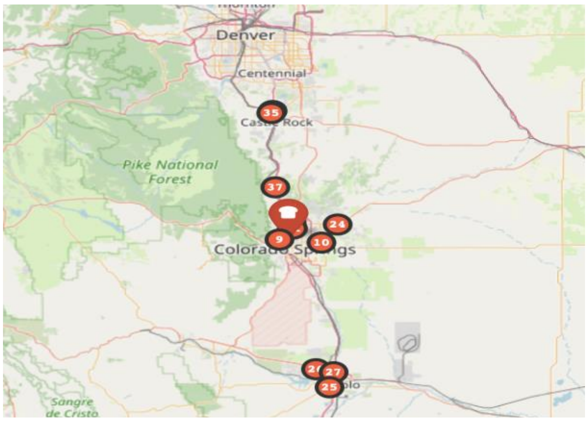


## ➢ Suggested Route 2

| Delivery Points | Deliveries Per Week | Weekly Mileage | Weekly Duration |
|---|---|---|---|
| 24 | 72 | 1269.03 | 44.53 |
| Delivery Points | Deliveries | Miles | Hours |

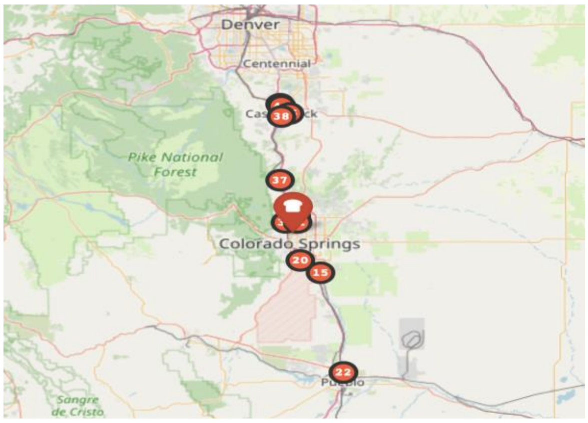
**Monday**

0 → 9 → 10 → 24 → 25 → 27 → 26 → 31 → 41 → 35 → 51 → 37 → 44 → 48 → 52 → 0



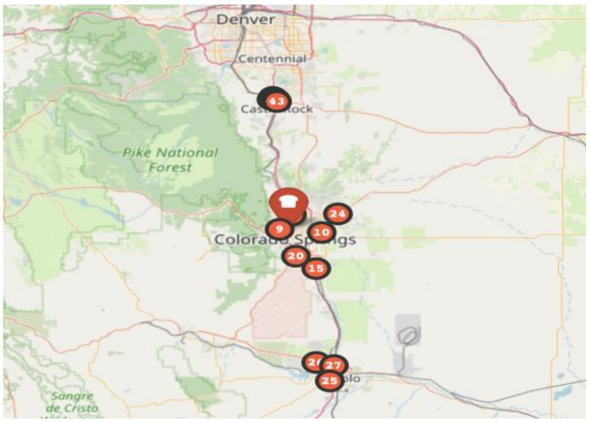

**Tuesday**

0 → 39 → 5 → 22 → 15 → 20 → 31 → 41 → 36 → 38 → 37 → 44 → 40 → 50 → 43 → 0



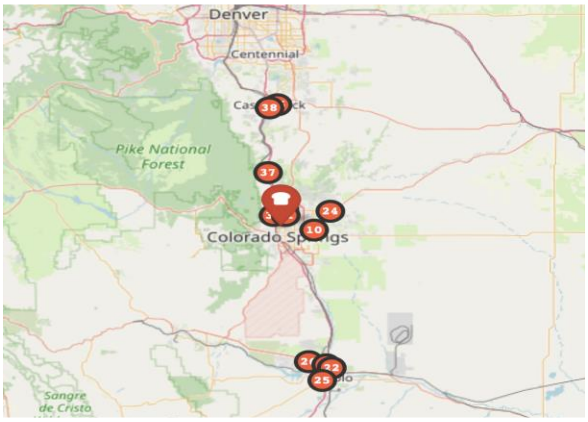

**Thursday**

0 → 9 → 10 → 24 → 15 → 20 → 25 → 27 → 26 → 35 → 51 → 40 → 50 → 43 → 48 → 52 → 0



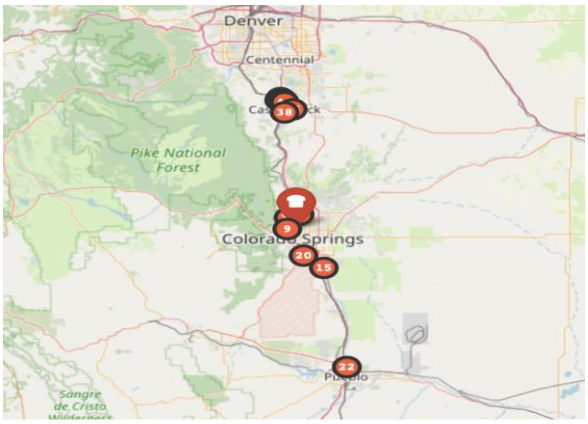

**Friday**

0 → 39 → 5 → 22 → 10 → 24 → 25 → 27 → 26 → 31 → 41 → 36 → 38 → 37 → 44 → 0




**Saturday**

0 → 9 → 5 → 22 → 39 → 15 → 20 → 35 → 51 → 36 → 38 → 40 → 50 → 43 → 48 → 52 → 0

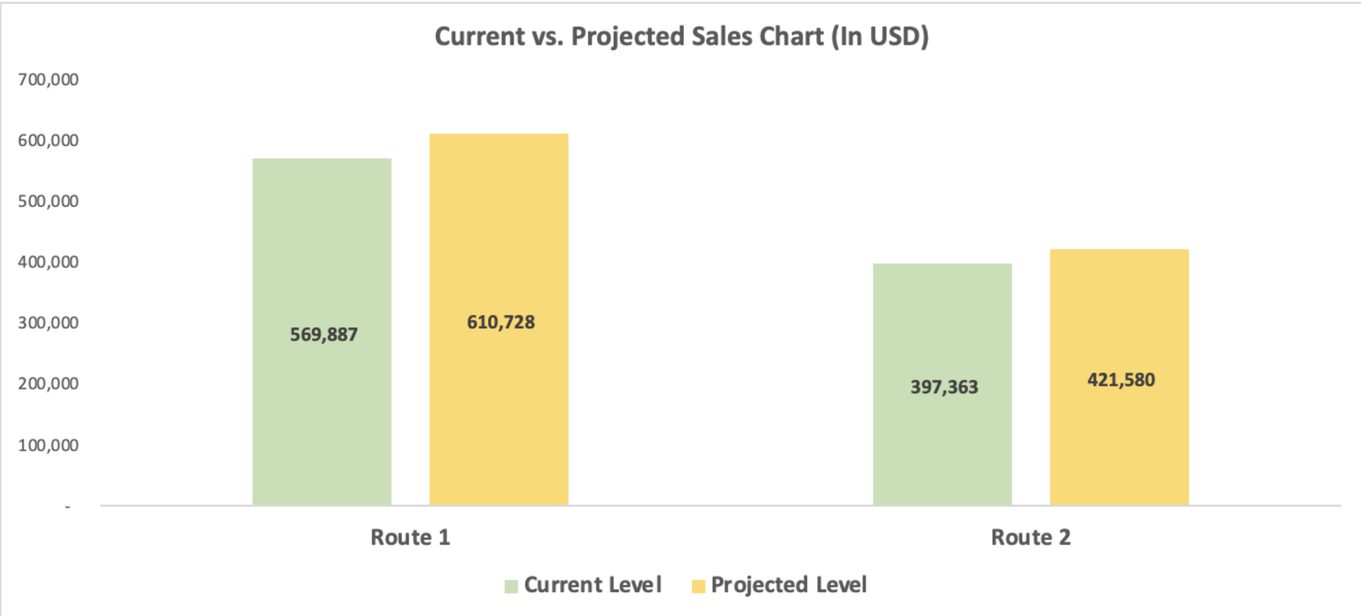


## ➢ Justification of Suggested Routes

We meticulously planned our delivery routes to optimize operations, with the primary goal of saving travel time, as outlined in our report. This strategy significantly reduces our overall operational expenses, including fuel costs. By employing a clustering algorithm, we simplified the complex issue of routing. This method ensured an equitable distribution of workload across all routes and simplified management by grouping retailers with similar total delivery time.

To maximize efficiency, our strategy incorporated several operational constraints. For example, we ensured that no store was visited more than three times within a three-day period, adhered to the designated working hours of delivery operations, and prioritized completing orders from specific stores when necessary. Such streamlined planning led to the optimal distribution of travel times across all suggested routes. Notably, Route 1's total travel time was 44.14 hours, while Route 2's was slightly longer, at 44.53 hours.

This fair distribution of time is crucial for minimizing driver idle time and promoting more consistent delivery schedules. Consequently, these carefully devised routes enhance service levels by ensuring timely and reliable deliveries to clients. This detailed explanation underscores the thoughtful consideration that went into our routing decisions, highlighting our dedication to operational improvement and superior customer service.




The graph indicates a significant revenue increase for both Route 1 and Route 2. Currently, Route 1's revenue stands at 569,887, with an anticipated increase of 40,841, projected to reach a total of 610,728. This represents an approximate 7.17% rise, signifying a notable surge in usage or preference for this specific route. Similarly, Route 2's revenue is currently at 397,363, with forecasts predicting an increase to 421,580, amounting to a change of 24,217 or a 6.09% increase. These figures reflect not only a general growth in revenue but also potential shifts in route preferences or improvements in route capacities.

## Future Directions

Unique strategic routes are being carved out to continuously improve and elevate Rudi's Bakery's delivery operations. These pathways aim to foster innovation within the company's logistical framework while staying true to its core principles and objectives. These future-focused strategies are intended to align with ongoing optimization efforts, ensuring a coherent and forward-thinking development of Rudi's delivery system.

a) **Improved Route Flexibility:** We are exploring the creation of an adaptive routing system capable of instantly recalculating routes in response to external factors such as adverse weather conditions, road closures, and traffic congestion. This flexibility will safeguard against delays and ensure the consistent reliability of deliveries.

b) **Direct Feedback Mechanisms**: Establish direct communication channels with retail partners to gather immediate feedback on delivery schedules and product reception. Leveraging this data to refine delivery operations will enhance partner satisfaction and foster opportunities for collaborative growth and adjustment strategies.

c) **Predictive Delivery Demand Modeling**: Utilizing machine learning algorithms, this approach forecasts future delivery needs by analysing sales data patterns, seasonal variations, and market trends. This proactive stance enables Rudi's to anticipate and meet evolving demand, ensuring optimal stock levels and minimizing waste.

d) **Green Logistics Initiatives**: Initiate a comprehensive review of delivery processes to lower carbon emissions. As part of a broader environmental strategy, this may include investing in renewable energy sources for logistic operations, experimenting with electric delivery vehicles, and optimizing routes for maximum fuel efficiency.

e) **Modular Expansion Strategy**: Develop a modular approach to route optimization that can dynamically adjust to business growth, including entering new markets or expansions. This strategy ensures the delivery model remains robust and adaptable as Rudi's geographic footprint and market presence expand.

By pursuing these future recommendations, Rudi's Bakery will not only refine its delivery operations but also reinforces its dedication to innovation, sustainability, and customer satisfaction. These initiatives are designed to ensure that as the organization progresses, it does so with an eye towards immediate efficiencies and the broader implications of its delivery operations, establishing a strong foundation for sustainable growth and operational excellence.

# Appendix

**A. Python Code for Optimization Model Development and Analysis**

**Google Colab: S24_004_Group_4_Project_1_Rudi’s_Optimization_Model.ipynb**

This appendix contains the full Python notebook with code used for developing the optimization model. Detailed comments within the notebook guide readers through each step of the computational analysis.

**B. Dataset Employed in Optimization Model**

**Data File: S24_004_Group_4_Project_1_Rudi’s_Delivery_Points.csv**

Included here is the CSV file that served as the primary dataset for this project.

**C. Dashboard for Route 1 and Route 2**

**PowerPoint Presentation: S24_004_Group_4_Project_1_Rudi's_Live_Dashboard.pptx**

The live dashboard provides tracking of routes and stores visited on specific days of the week. Additionally, by hovering over a store's name, users can view the delivery order number for that particular store.