**Document: Analytical Project - Optimizing Shipping Routes for a US Candy Distributor**

**Problem Statement**

The US Candy Distributor wanted to optimize its shipping routes from factories to customers across the United States and Canada. The primary goal was to identify:

1. The **most efficient shipping routes** from factories to customer cities.
2. The **least efficient routes** that required optimization.
3. Which products should be shifted to different factories to minimize delivery distances and costs.
4. Insights into product lines with the **best profit margins** to prioritize production and sales efforts.

**Data Sourced/Available**

**Tables and Fields:**

1. **Sales Table**:
   * Fields: Customer ID, City, Product ID, Division, Product Name, Sales, Gross Profit, Cost, Units.
2. **US Zips Table**:
   * Fields: City, Latitude (lat), Longitude (lng).
3. **Factories Table**:
   * Fields: Factory Name, Latitude, Longitude.
4. **Products Table**:
   * Fields: Division, Product Name, Factory, Unit Price, Unit Cost.
5. **Targets Table**:
   * Fields: Division, Target.

Additional cities' geolocation data were missing in the dataset and were manually inserted:

INSERT INTO US\_Zips (city, lat, lng)

VALUES

('New York City', 40.712776, -74.005974),

('Winnipeg', 49.895077, -97.138451),

... (and others as required).

**Approach**

1. **Create the Base Table:**

A new table, CustomerFactoryDistances, was created to store distances between customer locations and factories, along with the current delivering factory and the shortest distance.

CREATE TABLE CustomerFactoryDistances (

Customer\_ID VARCHAR(255),

City VARCHAR(255),

Product\_ID VARCHAR(255),

Current\_Delivering\_Factory VARCHAR(255),

Lots\_O\_Nuts FLOAT,

Wicked\_Choccy FLOAT,

Sugar\_Shack FLOAT,

Secret\_Factory FLOAT,

The\_Other\_Factory FLOAT,

min\_fact\_distance FLOAT,

min\_dist\_factory VARCHAR(255),

shift\_reqrd CHAR(1)

);

1. **Insert Base Data:**

Populated CustomerFactoryDistances with customer and product data:

INSERT INTO CustomerFactoryDistances (Customer\_ID, City, Product\_ID, Current\_Delivering\_Factory)

SELECT

s.Customer\_ID, s.City, s.Product\_ID, p.Factory

FROM candy\_sales s

JOIN candy\_products p ON s.Product\_ID = p.Product\_ID;

1. **Calculate Factory-to-Customer Distances:**

Using the Haversine formula, calculated the great-circle distance between factories and customer cities for each factory:

UPDATE CustomerFactoryDistances c

JOIN US\_Zips z ON c.City = z.city

JOIN candy\_factories f ON f.factory = 'Lots\_O\_Nuts'

SET c.Lots\_O\_Nuts = 6371 \* ACOS(

COS(RADIANS(z.lat)) \* COS(RADIANS(f.latitude)) \*

COS(RADIANS(f.longitude) - RADIANS(z.lng)) +

SIN(RADIANS(z.lat)) \* SIN(RADIANS(f.latitude))

);

Repeated this process for all factories.

1. **Find Minimum Distance and Factory:**

Identified the closest factory for each customer and marked it as the optimal factory.

UPDATE CustomerFactoryDistances

SET min\_fact\_distance = LEAST(Lots\_O\_Nuts, Wicked\_Choccy, Sugar\_Shack, Secret\_Factory, The\_Other\_Factory),

min\_dist\_factory = CASE

WHEN min\_fact\_distance = Lots\_O\_Nuts THEN 'Lots\_O\_Nuts'

WHEN min\_fact\_distance = Wicked\_Choccy THEN 'Wicked\_Choccy'

... (other factories).

END;

1. **Shift Requirement:**

Determined whether products should be shifted to a different factory for shorter delivery distances:

UPDATE CustomerFactoryDistances

SET shift\_reqrd = CASE

WHEN Current\_Delivering\_Factory = min\_dist\_factory THEN 0

ELSE 1

END;

1. **Product Margins:**

Calculated product margins as:

ALTER TABLE candy\_sales ADD COLUMN prod\_margin DECIMAL(10, 2);

UPDATE candy\_sales

SET prod\_margin = (Gross\_Profit / Sales) \* 100;

**Findings**

**1. Most Efficient Routes:**

Identified customer cities already receiving shipments from the closest factories:

SELECT DISTINCT City AS Customer\_Location, min\_dist\_factory AS Factory, min\_fact\_distance AS Distance\_in\_kms

FROM CustomerFactoryDistances

WHERE shift\_reqrd = 0

ORDER BY Distance\_in\_kms ASC;

**2. Least Efficient Routes:**

Highlighted customer cities requiring optimization:

SELECT DISTINCT City AS Customer\_Location, Current\_Delivering\_Factory AS Factory, Curr\_Fact\_Dist AS Distance\_in\_kms

FROM CustomerFactoryDistances

WHERE shift\_reqrd = 1

ORDER BY Distance\_in\_kms DESC;

**3. Best Product Margins:**

Found product lines with the highest margins:

SELECT DISTINCT p.Division AS Product\_Line, s.Product\_Name, s.prod\_margin AS Product\_Margin\_Percent

FROM candy\_sales s

JOIN candy\_products p ON s.Division = p.Division

ORDER BY Product\_Margin\_Percent DESC;

**4. Shift Recommendations:**

Suggested which product lines should be shifted to optimize shipping routes:

SELECT DISTINCT p.Division AS Product\_Line, p.Product\_Name, c.Current\_Delivering\_Factory, c.min\_dist\_factory AS Add\_To\_Factory

FROM CustomerFactoryDistances c

JOIN candy\_products p ON c.Product\_ID = p.Product\_ID

WHERE shift\_reqrd = 1;

**Actions Taken**

1. **Route Optimization:**
   * Adjusted factory assignments for products requiring a shift to minimize delivery distances.
2. **Production Prioritization:**
   * Focused on product lines with the highest margins for increased profitability.
3. **Customer Satisfaction:**
   * Reduced delivery times by optimizing routes, leading to improved customer experience.
4. **Cost Savings:**
   * Reduced shipping costs by assigning products to the closest factories.

**Analysis in Action**

Attached are SQL queries and their outputs, demonstrating the analysis process. The data highlights optimal shipping routes, product margins, and suggested shifts to achieve maximum efficiency.