```python
# Reese Koppel DoorDash Data Analysis (Jan 29, 2025)

import pandas as pd
import numpy as np
# Load and prepare the data
file_path = '~/code/data/doordash/Drive_Case_Study_Data_2024.csv'
# Error catching (for how DoorDash attempts to read the data)
try:
    data = pd.read_csv(file_path)
    data.info()
except FileNotFoundError:
    print(f"Error: The file at {file_path} was not found.")
    exit()
except Exception as e:
    print(f"Error reading the file: {e}")
    exit()
data = pd.read_csv(file_path)
# %%

# 1. Selecting Dashers for Widget Delivery Program

# Data pre-processing: We will drop the empty CANCELLED_AT column, convert time-
related columns to datetime format,
# fill missing values for numerical columns with 0, fill missing categorical values
with 'Unknown',
# and create a new feature for actual delivery duration in minutes.

# Drop completely empty column
df_cleaned = data.drop(columns=["CANCELLED_AT"])

# Convert time-related columns to datetime format
time_columns = [
    "CREATED_AT", "QUOTED_DELIVERY_TIME", "ESTIMATED_DELIVERY_TIME",
    "ACTUAL_PICKUP_TIME", "ACTUAL_DELIVERY_TIME", "DASHER_ASSIGNED_TIME",
    "DASHER_CONFIRMED_TIME", "DASHER_AT_STORE_TIME",
    "ACTUAL_PICKUP_TIME_GALAXY_A", "ACTUAL_DELIVERY_TIME_GALAXY_A",
    "DASHER_ASSIGNED_TIME_GALAXY_A", "DASHER_AT_STORE_TIME_GALAXY_A"
]

for col in time_columns:
    df_cleaned[col] = pd.to_datetime(df_cleaned[col], errors='coerce')

# Fill missing values for numerical columns with 0
num_cols = df_cleaned.select_dtypes(include=['float64']).columns
df_cleaned[num_cols] = df_cleaned[num_cols].fillna(0)

# Fill missing categorical values with 'Unknown'
cat_cols = df_cleaned.select_dtypes(include=['object']).columns
```

```
df_cleaned[cat_cols] = df_cleaned[cat_cols].fillna('Unknown')

# Create new feature: Actual delivery duration in minutes
df_cleaned["ACTUAL_DELIVERY_DURATION"] = (df_cleaned["ACTUAL_DELIVERY_TIME"] -
df_cleaned["ACTUAL_PICKUP_TIME"]).dt.total_seconds() / 60

# Display cleaned data info and first few rows
df_cleaned.info(), df_cleaned.head()
# %%
"""
Dasher Selection Process for Widget Deliveries

Since widget deliveries require higher precision and reliability, we should select
Dashers based on key performance indicators.
Given the differences between Dashattan (dense urban) and Doorlanta (sprawling
suburban), we will adjust criteria accordingly.
1. Selection Criteria for Dashers

    General Requirements (Both Cities)
        Minimum composite star rating of 4.5+ (higher-rated Dashers are more
reliable).
        Minimum of 500+ lifetime deliveries (ensures experience).
        Above-median on-time delivery percentage.

    MARKET_NAME-Specific Adjustments
        Dashattan (Urban)
            Prioritize bikers and motorbikes over cars for faster navigation in
traffic.
            Focus on Dashers with high short-distance delivery efficiency.
        Doorlanta (Suburban)
            Prioritize Dashers with cars due to longer distances.
            Consider experience in handling long-distance deliveries efficiently. """

# %%

# Reassign data to df
df = df_cleaned

# Convert timestamps to datetime
time_cols = ["CREATED_AT", "ACTUAL_PICKUP_TIME", "ACTUAL_DELIVERY_TIME"]
for col in time_cols:
    df[col] = pd.to_datetime(df[col], errors='coerce')

# Compute actual delivery duration in minutes
df["ACTUAL_DELIVERY_DURATION"] = (df["ACTUAL_DELIVERY_TIME"] -
df["ACTUAL_PICKUP_TIME"]).dt.total_seconds() / 60

# Compute proportion of deliveries which are on time
```

```python
df["ON_TIME_DELIVERY_RATE"] = df["NUM_ON_TIME_DELIVERIES"] / df["NUM_DELIVERIES"]

# Fill missing values
df.fillna({
    "COMPOSITE_STAR_RATING": df["COMPOSITE_STAR_RATING"].median(),  # Use median for numerical
    "NUM_DELIVERIES": df["NUM_DELIVERIES"].median(),
    "DASHER_VEHICLE_TYPE": "Unknown",  # Use 'Unknown' for categorical
}, inplace=True)
# First, how many drivers do we need?
# Assuming the other drivers do not mind potentially higher demand, we could denote some drivers as responsible for only widgets.
# We also assume that all of these drivers will want to do the more careful, tedious, laborious widget deliveries.
# Finally, we assume that all the bikers have containers that can protect widgets (as widgets are more fragile than food).
# If widget demand is 35% of total food delivery demand, that means we need 35% of the Dashers that we normally have.
# Let's calculate the number of Dashers we need for widget deliveries in each city.


# Total number of Dashers per city for regular deliveries

dashattan_dashers = df[df["MARKET_NAME"] == "Dashattan"]
doorlanta_dashers = df[df["MARKET_NAME"] == "Doorlanta"]

dashattan_count = dashattan_dashers["DASHER"].nunique()
doorlanta_count = doorlanta_dashers["DASHER"].nunique()

dashattan_count, doorlanta_count

# Interesting - it seems there are only 22 Dashers in Dashattan and 27 Dashers in Doorlanta that meet the criteria.

# So for widget deliveries, we would need how many Dashers in each city?

dashattan_need = int(0.35 * dashattan_count)
doorlanta_need = int(0.35 * doorlanta_count)

dashattan_need, doorlanta_need

# It seems we need 8 Dashers in Dashattan and 10 Dashers in Doorlanta for widget deliveries.
# Let's see how many high-performing Dashers we have in each city.

# Filtering high-performing Dashers based on defined criteria
high_performing_dashers = df[
    (df["COMPOSITE_STAR_RATING"] >= 4.5) &
    (df["NUM_DELIVERIES"] >= 500) &
    (df["ON_TIME_DELIVERY_RATE"] > df["ON_TIME_DELIVERY_RATE"].median())
    ]

# Splitting by city (MARKET_NAME)
dashattan_dashers = high_performing_dashers[high_performing_dashers["MARKET_NAME"] ==
"Dashattan"]
doorlanta_dashers = high_performing_dashers[high_performing_dashers["MARKET_NAME"] ==
"Doorlanta"]

# Count selected Dashers per city (MARKET_NAME)
dashattan_count = dashattan_dashers["DASHER"].nunique()
doorlanta_count = doorlanta_dashers["DASHER"].nunique()

dashattan_count, doorlanta_count

# The number of selected Dashers for Dashattan is 3, and for Doorlanta is 11.
# We need to loosen up the criteria to find more Dashers in Dashattan.
# %%
decently_performing_dashers = df[
    (df["COMPOSITE_STAR_RATING"] >= 4.5) &
    (df["NUM_DELIVERIES"] >= 500) &
    (df["ON_TIME_DELIVERY_RATE"] > 0.02)
    ]
# Splitting by city (MARKET_NAME)
dashattan_dashers =
decently_performing_dashers[decently_performing_dashers["MARKET_NAME"] == "Dashattan"]

# Count selected Dashers per city (MARKET_NAME)
dashattan_count = dashattan_dashers["DASHER"].nunique()

dashattan_count

# The number of selected Dashers for Dashattan is up to 4. Can we get towards 8? Let's
see if we can loosen the criteria further.

# %%
dashattan_final_search = df[
    (df["COMPOSITE_STAR_RATING"] >= 4) &
    (df["NUM_DELIVERIES"] >= 400) &
    (df["ON_TIME_DELIVERY_RATE"] > 0.02) &
    (df["MARKET_NAME"] == "Dashattan")
    ]

# Count selected Dashers per city (MARKET_NAME)
dashattan_count = dashattan_final_search["DASHER"].nunique()

dashattan_count
```

```
# The number of selected Dashers for Dashattan is up to 7. While we were targeting 8, this is a good number to start with.
# We can always adjust the criteria based on performance, workload management, and demand.
# We do not want to relax the standards too far as the widgets are valuable and require careful handling.

# Reassign the final Dashattan Dashers
dashattan_dashers = dashattan_final_search

# Should the criteria be the same for both cities? Considerations include bikes versus cars.
# %%
print(dashattan_dashers.drop_duplicates(subset=["DASHER"]))
print(dashattan_dashers.drop_duplicates(subset=["VEHICLE"]))
# 4 Dashattan Dashers are using bicycles, and 3 are using cars.
print(doorlanta_dashers.drop_duplicates(subset=["DASHER"]))
print(doorlanta_dashers.drop_duplicates(subset=["VEHICLE"]))
# All 11 Dashlanta Dashers are using cars.
# We are okay with some Dashattan Dashers using bicycles and some using cars.
# While bicycles can navigate traffic faster, they might not be the best for our fragile widgets.
# Therefore, we will start with both cars and bicycles but will monitor closely to reassess if needed.
# We are okay with all Dashlanta Dashers being cars, as they are necessary for handling long distances.
# %%
# 2. Widget Satchel Distribution Plan
"""
In Dashattan:
Implement centralized pickup locations strategically placed near high-demand zones to maximize efficiency.
Dashers should collect satchels at the beginning of their shifts to ensure they are ready for widget deliveries.
In Doorlanta:
Due to the city’s sprawling nature, distribute satchels directly to the homes of the select Dashers delivering exclusively widgets.
If mailing satchels is cost-prohibitive, consider leveraging regional distribution hubs to manage logistics more effectively.
"""

# 3. Measuring Program Success
"""
    Delivery Quality
        On-time delivery rate (percentage of widget orders delivered within a margin of error of the expected delivery time).
        Delivery accuracy (percentage of orders delivered correctly without damage).
    Dasher Performance
        Average delivery duration (time from pickup to drop-off).
        Dasher ratings on widget deliveries.

    DoorDash Success (Profitability & Efficiency)
        Profit margin per widget delivery (revenue - operational costs).
        Average cost per delivery (including satchel logistics, Dasher incentives).

    Merchant Success
        Merchant rating of delivery service (feedback on timeliness, condition).
        Merchant retention rate (percentage of merchants continuing to use DoorDash
for widget deliveries).
        Delivery success rate (percentage of widget deliveries completed without
issues).

    Customer Satisfaction
        Customer star ratings for widget deliveries.
        Customer complaints per 100 deliveries (damaged/missing widgets). """
```