Import Libraries

In [205]:
```
import numpy as np
import pandas as pd
from datetime import timedelta
from matplotlib import pyplot as plt
import seaborn as sns

import warnings
# Ignore all warnings
warnings.filterwarnings("ignore")
```

Context:

A large e-commerce company has contracted us to provide delivery services from several of their warehouses(or sites) to customers' doorsteps at several locations across the country.

The dataset contains information about the performance of our delivery agents working at our client's sites across cities in India.

The idea of this exercise is to gain insights into their performance and make data driven strategic recommendations for imporvement.

NOTE: The dataset contains information for only one month, i.e June 2023

Features in the Dataset

- Site Code - A unique ID belonging to the site at which our delivery agents work.
- City - Name of the city in which the site is located.
- Vehicle Type - "Van DCD" means the delivery agent drives his own van and delivers packages. "Bike" means the delivery agent rides his own bike and delivers packages. "Van D+DA" means the delivery agent doesn't drive, but is instead driven around the city by a van driver.
- Cluster - Cities are grouped into clusters based on certain characteristics.
- Date - The date that the delivery agent reported for work.
- Delivery Agent ID - A unique identifier assigned to each delivery agent who works with us.
- Shift - "A" indicates morning, "B" indicates afternoon, "C" indicates evening.
- Unsuccessful_Attempts - The number of packages that the delivery agent attempted to deliver, but the delivery was not possible for various reasons
- Process_Deviations - The number of times the delivery agent deviated from the process during the shift.
- Delivered - The number of packages that the delivery agent delivered during the shift.
- Customer_Rejects - The number of packages that customers rejected when the delivery agent went to deliver them.
- Picked_up_Customer_Returns - The number of packages the delivery agent picked up

from customers who wanted to return something that they had ordered earlier.
- Picked_up_Seller - The number of packages that the delivery agent picked up from sellers on the platform. These packages would be sent out for delivery to customers the next day.
- billing_amt - The amount that we bill our clients for the services rendered by our delivery agents.

Question 1

Load the dataset into a dataframe and name it "daily_df". Display a sample of 10 random rows, display the shape of the dataset, the datatypes of each column, and check for missing values.

In [206]:
```
# We have imported the dataset via under daily_df dataframe
daily_df=pd.read_excel("Python Ninja - Advanced - Dataset.xlsx")
#In the above code we have display the 10 random rows for our dataframe
daily_df.head(10)
```

Out[206]:

	Site Code	city	Vehicle_type	cluster	Date	Delivery Agent ID	Shift	Unsuccessful_Attempt
0	Site_22	HYDERABAD	Bike	4	2023-06-27	DA_280	A	
1	Site_10	BANGALORE	Van DCD	4	2023-06-08	DA_273	A	
2	Site_3	BANGALORE	Bike	4	2023-06-25	DA_370	C	
3	Site_10	BANGALORE	Van DCD	4	2023-06-22	DA_118	B	
4	Site_4	BANGALORE	Van DCD	4	2023-06-11	DA_136	B	
5	Site_3	BANGALORE	Van DCD	4	2023-06-24	DA_36	B	
6	Site_19	HYDERABAD	Bike	4	2023-06-05	DA_295	C	
7	Site_27	MUMBAI	Bike	3	2023-06-08	DA_224	A	
8	Site_23	CHENNAI	Bike	3	2023-06-23	DA_686	A	
9	Site_16	BANGALORE	Bike	4	2023-06-23	DA_569	B	

In [207]:
```
# Display the shape of the dataset which is 11279 rows and 14 column
daily_df.shape
# Display the data type of each column
daily_df.dtypes
```

Out[207]:
```
Site Code                          object
city                               object
Vehicle_type                       object
cluster                             int64
Date                       datetime64[ns]
Delivery Agent ID                  object
Shift                              object
Unsuccessful_Attempts               int64
Process_Deviations                  int64
Delivered                           int64
Customer_Rejects                    int64
Picked_up_Customer_Returns          int64
Picked up_Seller                    int64
billing_amt                         int64
dtype: object
```

In [208]:
```
# Now display the count of missing value in each column
daily_df.isna().sum()
# After evaluating the dataframe we found that there are no any missing value
```

Out[208]:
```
Site Code                    0
city                         0
Vehicle_type                 0
cluster                      0
Date                         0
Delivery Agent ID            0
Shift                        0
Unsuccessful_Attempts        0
Process_Deviations           0
Delivered                    0
Customer_Rejects             0
Picked_up_Customer_Returns   0
Picked up_Seller             0
billing_amt                  0
dtype: int64
```

Question 2

Part-A

You will notice that there are delivery agents who have used different vehicle types in different shifts.

Find what vehicle type was used for majority of the shifts, then overwrite all the other minority values with the majority values.

For example - if DA_1 used a bike in 20 shifts and used a van in 2 shifts, you need to make

sure that the "Vehicle Type" for all 22 shifts is set to "Bike" as that was used for majority of the shifts

In [ ]:
```
'''Now we can group by the dataset with Delivery agent id and shift and
then use a lambda function with value_counts() to calculate the most frequent
combination of delivery agent id and then store it into the new dataframe whic

vehicle_type_majority=pd.DataFrame(daily_df.groupby(["Delivery Agent ID","Shif
```

In [ ]:
```
''' We merge the original daily_df dataframe with the new vehicle_type_majorit
Now we updated the vehicle type column with the majority column values and dro

daily_df=daily_df.merge(vehicle_type_majority,on=["Delivery Agent ID","Shift"]
```

Part-B:

Total assigned packages can be calculated using the following formula:

Total Assigned = Unsuccessful_Attempts + Delivered + Customer_Rejects + Picked_up_Customer_Returns + Picked up_Seller

Use this formula and add a new column in the dataframe called "Total Assigned"

In [ ]:
```
## We have now successfully added the Total assigned package column by given t
daily_df["Total Assigned"]=daily_df["Unsuccessful_Attempts"]+daily_df["Deliver
```

In [ ]:
```
daily_df
```

Part-C:

"Productivity" is a metric that aims at measuring how much work a delivery agent is doing during his shift. This can later be compared with the rest of the workforce to identify delivery agents who are both - outperforming as well as under performing.

It can be calculated using the following formula:

Productivity = Delivered + Customer_Rejects + Picked_up_Customer_Returns + Picked up_Seller.

Calculate "Productivity" and show it in a new column in the daily_df.

In [ ]:
```
# We have now successfully added the Productivity column by given the above fo
daily_df["Productivity"]=daily_df["Delivered"]+daily_df["Customer_Rejects"]+da
```

Part-D:

Analyze the data and report the ranges of productivity that different vehicle types have in different types of shifts. What are your observations?

In [ ]:
```
# For the above formula we have grouped the vehicle type and shift as per the
# and put it in a new dataframe called productivity_ranges
productivity_ranges=daily_df.groupby(["Vehicle_type","Shift",])["Productivity"
productivity_ranges
```

In [ ]:
```
''' Now here we have visualize the productivity as per the below code
1. We have created a loop shift in which we have included the unique value of
2. Now we have calculated the range for productivity of max and min for each g
3.we  calculated productivity ranges for different vehicle types and shifts.
4. After ward we have visualize the dataset with each shift having its own set
productivity'''

plt.figure(figsize=(10, 6))
for shift in productivity_ranges['Shift'].unique():
    shift_data = productivity_ranges[productivity_ranges['Shift'] == shift]
    plt.bar(shift_data['Vehicle_type'], shift_data['max'] - shift_data['min'],
plt.xlabel('Vehicle Type')
plt.ylabel('Productivity Range')
plt.title('Productivity Range by Vehicle Type and Shift')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Observation

1. Van D+DA vehicle type has the lowest productivity range however Van DCD has the highest productivity range .
2. Most of the Delivery agent prefer bike in evening shift . However most of the people prefer VAN DCD in morning shift.
3. In B which is afternoon shift also most of the delivery agent used Van DCD vehicle type

Part E:

Low productivity adversely affects both - service levels as well as unit level economics of the business. Curbing this is quintessential to running a good operation.

Now that you have a sense of the productivity levels of different vehicle types in each shift, use this information to formulate logic to classify all the samples across all shifts as either "productivity-ok" or "productivity-low". Add this information into a column called "Productivity Category".

- Keep in mind that the productivity of one vehicle type is not comparable to the other. A bike may travel faster when there is more traffic. A van may be able to carry a much larger number of packages, etc...
- Keep in mind that productivity from Shift A is not comparable to productivity from Shifts B

or C or vice versa. Each shift should have it's own threshold for defining low productivity.
- Most importantly, keep in mind that the ideal solution would be to identify a smaller set of people who contribute to the largest part of the problem.
- You are not being given any specific formula here to do this. You will need to think creatively and also give an explanation of your method and why you chose to do things that way.

In [ ]:
```
# 1. We have created a grouped dataframe in which we basically provided the gr
# will be using on later part
grouped = daily_df.groupby(["Vehicle_type", "Shift"])

# 2.Now we have created an empty dataframe to store the result
productivity_categories = []

# 3.This loop iterates through each group created by the groupby operation. gr
#and "Shift" for the current group, while group_data contains the actual data
for group_key, group_data in grouped:
    Vehicle_type, Shift = group_key
    group_productivity_values = group_data["Productivity"].values

    # Calculate the threshold based on the median
    threshold = group_data["Productivity"].median()

    # Classify samples as "productivity-ok" or "productivity-low"
    category = ["productivity-ok" if value >= threshold else "productivity-low
    productivity_categories.extend(category)

# Add the "Productivity Category" column to the DataFrame
daily_df["Productivity Category"] = productivity_categories
daily_df
```

In [ ]:
```
''' In summary, this code groups the data by "Vehicle Type" and "Shift," calcu
    value within each group, classifies samples within each group based on the
    (Threshold-  we have taken the median value for each group ) ,
    and adds the classification results as a new column in the DataFrame. This
    the product category value as per the median for each grouped set of the c
```

Question 3

Part A:

The daily_df contains information of how each delivery agent performed on each day that they reported in June 2023. Create a new dataframe that summarizes the performance of each delivery agent for the entire month of June 2023, using the data from the daily_df.

Name this new dataframe "monthly_df".

The new dataframe should contain the following columns:

- Delivery Agent ID
- Vehicle Type

- Cluster
- City
- Site Code
- Shifts Worked
- Total Assigned
- Unsuccessful_Attempts
- Process_Deviations
- Productivity
- billing_amt

In [ ]:
```
## We have created a variable column in which we have included all the column
column=['Delivery Agent ID','Vehicle_type','cluster','city','Site Code','Total
```

In [ ]:
```
'''We have created a group by dataframe called N_F in which we
have grouped the count for each delivery agent in respect for their count of s
N_F=daily_df.groupby(['Delivery Agent ID'])['Shift'].value_counts().reset_inde
```

In [ ]:
```
N_F=N_F.groupby(['Delivery Agent ID'])['count'].sum().reset_index()
N_F
```

In [ ]:
```
## We have merged the count value in our original dataframe
daily_df=daily_df.merge(N_F,on='Delivery Agent ID',how='left')
```

In [ ]:
```
## We have replace the name now for our newly assigned column from count to Sh
daily_df.rename(columns={'count':'Shift_Worked'},inplace=True)
```

In [ ]:
```
monthly_df=daily_df[column]

monthly_df
```

In [ ]:
```
''' We have also evaluate the performance of each delivery agent with the foll
Performance = daily_df.groupby('Delivery Agent ID').agg({
    'Delivered': 'sum',
    'Picked up_Seller':'sum',
    'Customer_Rejects': 'sum',
    'Picked_up_Customer_Returns': 'sum',
    'Unsuccessful_Attempts': 'sum',
    'Process_Deviations': 'sum',
    'billing_amt': 'sum',
    'Shift_Worked':'sum'
}).reset_index()
Performance
```

Part B:

"Delivery Success Rate(DSR)" is a metric that measures the quality of a delivery agent's work. If an angent has low DSR, it means that he/she would have a higher number of unsuccessful

attempts.

DSR can be calculated using the following formula:

DSR = Productivity / Total Assigned.

Using this formula, calculate and add a new column called "DSR" in the monthly_df. Round the values to 2 decimal placecs

In [ ]:
```
# As per the above formula we have added a column called DSR and adding it int
monthly_df['DSR']=round(monthly_df["Productivity"]/monthly_df["Total Assigned"
monthly_df
```

Part C:

The productivity in the monthly_df is an aggregate of the whole month's productivity. Create a new column called "Avg_Productivity" that contains the average productivity per shift of each delivery agent. The values in this column should be expressed as integers.

Do the same with Process_Deviations. Calculate the average number of deviations per shift worked and put it into a column called "Avg_Deviations". Here the values should be rounded to 2 decimal places

In [ ]:
```
# Here we have added the two column and aggregate the average of productivity
# in new dataFrame called New_df
New_df=daily_df.groupby(["Delivery Agent ID","Shift"]).agg({"Productivity":"me
New_df.rename(columns={"Productivity":"Avg_Productivity","Process_Deviations":
```

In [ ]:
```
# Now we have converted avg_productivity column as integer and process deviati
New_df[["Avg_Productivity","Avg_Process_Deviation"]]=New_df[["Avg_Productivity
New_df["Avg_Productivity"]=New_df["Avg_Productivity"].astype('int')
New_df
```

Part D:

Examine the distributions of the features in the monthly_df in both forms - tabular and plotted.

All columns that contain categorical information (data that has no ordinal value) should be shown as bar graphs. All columns that contain numerical data (data with ordinal value) should be shown as distribution curves or histograms.

State your observations and inferences.

In [ ]:
```
# Here we have included all the categorical column which has been found on mon
Categorical_column= ['Vehicle_type','city','Site Code' ]
```

In [ ]:
```
# Now we have created a for loop so that we can provide a bar plot with the he
for col in Categorical_column:
    plt.figure(figsize=(10,6))
    sns.countplot(data=monthly_df,x=col)
    plt.title(f"Distribution of {col}")
    plt.xticks(rotation=45)
    plt.show()
```

In [ ]:
```
# Now we have created a variable in which we have stored all the numeric colum
numeric_column=["cluster","Total Assigned","Unsuccessful_Attempts","Process_De
```

In [ ]:
```
# Now we have created a for loop so that we can provide a hisogram plot with t
for col in numeric_column:
    plt.figure(figsize=(10,6))
    sns.histplot(data=monthly_df,x=col,bins=20,kde=True)
    plt.title(f"Distribution Of {col}")
    plt.show()
```

Observation

** Bar Graph

1. With the help of bar graph we found that most of the delivery agent has been using the bike vehicle and less no of people has been using the Van D +DA vehicle type
2. Most of the delivery agent had try to do the delivery in bengluru district and less number of delivery agent has been try to do delivery in chennai hence need to be more focus in chennai city
3. Site_22 has the highest number of distribution in bar graph however site_8 has the lowest number of distribution

** Histogram Graph

1. There are less number of delivery agent who have the total assigned delivery were more than 150 . The distribution of Total assigned delivery was arount between 1-100.
2. The histogram bar graph for total no. of shift lies between medium to high which can showcase that highest number of delivery agent was working maximum days in a month
3. Most of the delivery billed between 500-1500 Which mean we need to be very focused for those delivery who were lies in between of this price
4. The distribution of unsuccessfull attempt was 0-5 per delivery agent

Part-E

Now that you have understood the distributions of the data in the monthly_df, you need to:

1. Classify each agent on the basis of their DSR as either "DSR-ok" or "DSR-low"
2. Classify each agent on the basis of the number of deviations they've had as either "deviations-ok" or "deviations-high"

This classification should be represented in two new columns in the monthly_df called "DSR

Category" and "Deviations Category"

Note that DSR and Deviations are comparable across all vehicle types, all shifts and all geographical areas. For example - there is no valid reason for vans in Mumbai to have lower DSR or higher deviations when compared to bikes in Hyderabad or Banagalore or Pune. The same benchmark of quality applies to all vehicle types working all shifts in all cities.

Once again, you are not being given any explicit thresholds to use for this classification. You are expected to use your analysis to decide the thresholds.

An effective solution would be one which identifies a smaller group of people that contribute to the largest part of the problem.

In [87]:
```
DF1=daily_df["Shift"]
monthly_df["Shift"]=DF1
```

In [88]:
```
# 1. We have created a grouped dataframe in which we basically provided the gr
# will be using on later part
grouped_DSR = monthly_df.groupby(["Vehicle_type", "Shift","city"])

# 2.Now we have created an empty dataframe to store the result
DSR_categories = []

# 3.This loop iterates through each group created by the groupby operation. gr
# "Shift",and city for the current group, while group_data contains the actual
for DSR_key, group_DSR_data in grouped_DSR:
    Vehicle_type, Shift,city = DSR_key
    group_DSR_values = group_DSR_data["DSR"].values

    # Calculate the threshold based on the median
    threshold = group_DSR_data["DSR"].median()

    # Classify samples as "DSR-ok" or "DSR Low"
    category = ["DSR-ok" if value >= threshold else "DSR-low" for value in gro
    DSR_categories.extend(category)

# Add the "DSR Category" column to the DataFrame
monthly_df["DSR Category"] = DSR_categories
monthly_df
```

Out[88]:

	Delivery Agent ID	Vehicle_type	cluster	city	Site Code	Total Assigned	Unsuccessful_Attempts
0	DA_280	Bike	4	HYDERABAD	Site_22	30	0
1	DA_273	Van DCD	4	BANGALORE	Site_10	81	0
2	DA_370	Bike	4	BANGALORE	Site_3	19	2
3	DA_118	Van DCD	4	BANGALORE	Site_10	16	1
4	DA_136	Van DCD	4	BANGALORE	Site_4	46	5
...	...	...	...	...	...	...	...
11274	DA_377	Van DCD	4	BANGALORE	Site_10	42	0
11275	DA_351	Van D+DA	5	MUMBAI	Site_27	24	2
11276	DA_685	Van DCD	4	BANGALORE	Site_3	30	4
11277	DA_10	Van DCD	4	BANGALORE	Site_3	78	4
11278	DA_361	Bike	3	MUMBAI	Site_26	23	0

11279 rows × 14 columns

In [89]:
```
# 1. We have created a grouped dataframe in which we basically provided the gr
# will be using on later part
grouped_Deviation = monthly_df.groupby(["Vehicle_type", "Shift","city"])

# 2.Now we have created an empty dataframe to store the result
Deviation_categories = []

# 3.This loop iterates through each group created by the groupby operation. gr
# "Shift",and city for the current group, while group_data contains the actual
for Deviation_key, group_Deviation_data in grouped_Deviation:
    Vehicle_type, Shift,city = Deviation_key
    group_Deviation_values = group_Deviation_data["Process_Deviations"].values

    # Calculate the threshold based on the median
    threshold = group_Deviation_data["Process_Deviations"].median()

    # Classify samples as "deviation-ok" or "deviation Low"
    category = ["deviations-ok" if value >= threshold else "deviations-low" fo
    Deviation_categories.extend(category)

# Add the "Deviation Category" column to the DataFrame
monthly_df["Deviation Category"] = Deviation_categories
monthly_df
```

Out[89]:

	Delivery Agent ID	Vehicle_type	cluster	city	Site Code	Total Assigned	Unsuccessful_Attempts
0	DA_280	Bike	4	HYDERABAD	Site_22	30	0
1	DA_273	Van DCD	4	BANGALORE	Site_10	81	0
2	DA_370	Bike	4	BANGALORE	Site_3	19	2
3	DA_118	Van DCD	4	BANGALORE	Site_10	16	1
4	DA_136	Van DCD	4	BANGALORE	Site_4	46	5
...	...	...	...	...	...	...	...
11274	DA_377	Van DCD	4	BANGALORE	Site_10	42	0

Question 4

Part-A

Create a new dataframe that shows the number of A, B, and C shifts each delivery agent has done in the whole month. Also include in a separate column the total number of low productivity shifts each delivery agent has had in the whole month. Call this new dataframe "shifts_df". Make sure that there are no nan values in the new dataframe.

To be more clear, the new dataframe(shifts_df) should have the following columns:

- Delivery Agent ID
- Total no. of 'A' Shifts
- Total no. of 'B' Shifts
- Total no. of 'C' Shifts
- Total no. of low productivity shifts

In [90]:
```
# Now we have created a dataframe mn_df in which we have grouped by the column
mn_df=monthly_df.groupby(["Delivery Agent ID"])["Shift"].value_counts().reset_
```

In [91]:
```
# Now we have created the pivot table where we have created the total no of re
pivot_table = pd.pivot_table(mn_df, index='Delivery Agent ID', columns='Shift'
pivot_table.rename(columns={'A': 'Total no. of \'A\' Shifts', 'B': 'Total no.
```

In [92]:
```
# Now we have created a new dataframe called pivot_DF In whic we want to extra
pivot_df=daily_df[["Delivery Agent ID","Shift","Productivity Category"]]
pivot_df=pivot_df.groupby(["Delivery Agent ID","Productivity Category"])["Shif
```

In [93]:
```
# Now we again created a pivot table in which we have extracted the total coun
Pro=pivot_df.groupby(["Delivery Agent ID"])["Productivity Category"].value_cou
Pro=pd.pivot_table(Pro, index='Delivery Agent ID', columns='Productivity Categ
Pro
```

Out[93]:

Productivity Category	productivity-low	productivity-ok
Delivery Agent ID		
DA_1	3	2
DA_10	2	2
DA_100	1	1
DA_101	1	2
DA_102	1	1
...	...	...
DA_95	3	1
DA_96	2	2
DA_97	2	1
DA_98	1	1
DA_99	2	2

696 rows × 2 columns

In [94]:
```
# Now we have keep the productivity low column in our new dataframe called shi
pivot_table["Total no. of low productivity shifts"]=Pro["productivity-low"]
shifts_df=pivot_table
```

Part B

Merge the monthly_df and the shifts_df so that all the features are in one single dataframe. Call this new dataframe "final_df".

Then calculate and create a new column called "%_low_prod_shifts". For the values in this column divide the total number of low productivity shifts by the total no. of shifts worked and round the result to two decimal places.

In [95]:
```
shifts_df.reset_index(inplace=True)
```

In [96]:
```
final_df=monthly_df.merge(shifts_df,on="Delivery Agent ID",how="left")
```

In [97]:
```
final_df["%_low_prod_shifts"]=final_df["Total no. of low productivity shifts"]
```

In [98]:
```
final_df["%_low_prod_shifts"]=final_df["%_low_prod_shifts"].round(2)
```

In [99]:
```
final_df
```

Out[99]:

	Delivery Agent ID	Vehicle_type	cluster	city	Site Code	Total Assigned	Unsuccessful_Attempts
0	DA_280	Bike	4	HYDERABAD	Site_22	30	0
1	DA_273	Van DCD	4	BANGALORE	Site_10	81	0
2	DA_370	Bike	4	BANGALORE	Site_3	19	2
3	DA_118	Van DCD	4	BANGALORE	Site_10	16	1
4	DA_136	Van DCD	4	BANGALORE	Site_4	46	5
...	...	...	...	...	...	...	...
11274	DA_377	Van DCD	4	BANGALORE	Site_10	42	0
11275	DA_351	Van D+DA	5	MUMBAI	Site_27	24	2
11276	DA_685	Van DCD	4	BANGALORE	Site_3	30	4
11277	DA_10	Van DCD	4	BANGALORE	Site_3	78	4
11278	DA_361	Bike	3	MUMBAI	Site_26	23	0

11279 rows × 20 columns

Part C

Create three new columns - "%_shifts_A", "%_shifts_b", and "%_shifts_C".

If a delivery agent had a total of 10 shifts, and 5 out of them were Shift A, then "%_shifts_A" should be 0.5.

All values in the new columns should be rounded to 2 decimal places.

In [100]:
```
x= final_df["Shift_Worked"]
final_df["%_shifts_A"]=round(final_df["Total no. of 'A' Shifts"]/x,2)
final_df["%_shifts_B"]=round(final_df["Total no. of 'B' Shifts"]/x,2)
final_df["%_shifts_C"]=round(final_df["Total no. of 'C' Shifts"]/x,2)
```

Question 5

Part-A:

Irregularity:

When we say a delivery agent is irregular to work, we mean that they were associated with us for a certain period, however, during this period they were absent frequently.

For example - A delivery agent's first day of work was 5th June and the last day of work was 25th June. However during this period, the agent worked for only 10 out of the possible 20 days. We would then say that this delivery agent is "irregular".

Let's look at another scenario where a delivery agent has worked for only 5 out of the possible 30 days in the whole month, and those 5 days are towards the end of the month, it probably means that they joined us late, and hence cannot be classified as "irreguar", and they should ideally be classified as "new".

Keeping this context in mind, classify each delivery agent in the dataset as "regular" or "irregular" or "new". This classification should be shown in a new dataframe called "regularity_df". This new dataframe should contain only two columns - "Delivery Agent ID" and "Regularity Classification"

In this question, we are not explicity prescribing the logic nor the threshold of working days to be used for this classification. You are expected to get creative and take a calculated decision on how to go about this.

Once all the delivery agents have been classified, explain the logic and reasoning behind the method you have chosen.

In [104]:
```
'''We have created the new dataframe called days_worked in which we have group
and extracting the unique date worked for each Delivery agent  for those'''
days_worked = daily_df.groupby('Delivery Agent ID')['Date'].nunique().reset_in

# Now we have define the regular threshold for those who have 80% available an
regular_threshold = 0.8  # e.g., 80% of available days
irregular_threshold = 0.4  # e.g., 40% of available days

'''
Now we have created a function called classify_regularity in which if a person
the value came as regular  and if it greater than irregular threshold than val
def classify_regularity(row):
    if row['Days_Worked'] / total_days >= regular_threshold:
        return 'regular'
    elif row['Days_Worked'] / total_days >= irregular_threshold:
        return 'irregular'
    else:
        return 'new'

#  We have Calculated total days in the entire dataset
total_days = daily_df['Date'].nunique()

# Apply the classification logic
days_worked['Regularity Classification'] = days_worked.apply(classify_regulari

# Create the "regularity_df" DataFrame
regularity_df = days_worked[['Delivery Agent ID', 'Regularity Classification']



regularity_df
```

Out[104]:

	Delivery Agent ID	Regularity Classification
0	DA_1	irregular
1	DA_10	irregular
2	DA_100	new
3	DA_101	irregular
4	DA_102	regular
...	...	...
691	DA_95	regular
692	DA_96	irregular
693	DA_97	regular
694	DA_98	new
695	DA_99	irregular

696 rows × 2 columns

Part-B

Merge the regularity_df into the final_df so that all the features are available in one single dataframe

In [105]:
```
final_df=final_df.merge(regularity_df,on="Delivery Agent ID",how="left")
```

In [106]:
```
final_df
```

Out[106]:

	Delivery Agent ID	Vehicle_type	cluster	city	Site Code	Total Assigned	Unsuccessful_Attempts
0	DA_280	Bike	4	HYDERABAD	Site_22	30	0
1	DA_273	Van DCD	4	BANGALORE	Site_10	81	0
2	DA_370	Bike	4	BANGALORE	Site_3	19	2
3	DA_118	Van DCD	4	BANGALORE	Site_10	16	1
4	DA_136	Van DCD	4	BANGALORE	Site_4	46	5
...	...	...	...	...	...	...	...
11274	DA_377	Van DCD	4	BANGALORE	Site_10	42	0
11275	DA_351	Van D+DA	5	MUMBAI	Site_27	24	2
11276	DA_685	Van DCD	4	BANGALORE	Site_3	30	4
11277	DA_10	Van DCD	4	BANGALORE	Site_3	78	4
11278	DA_361	Bike	3	MUMBAI	Site_26	23	0

11279 rows × 24 columns

Question 6

Part-A

Keeping in mind that the final_df contains a mix of categorical and numerical variables, find out if any of the features are correlated to each other using an appropriate method. Find out and report if any variables have a strong positive or negative correlation.

Explain which method(s) you have chosen and why.

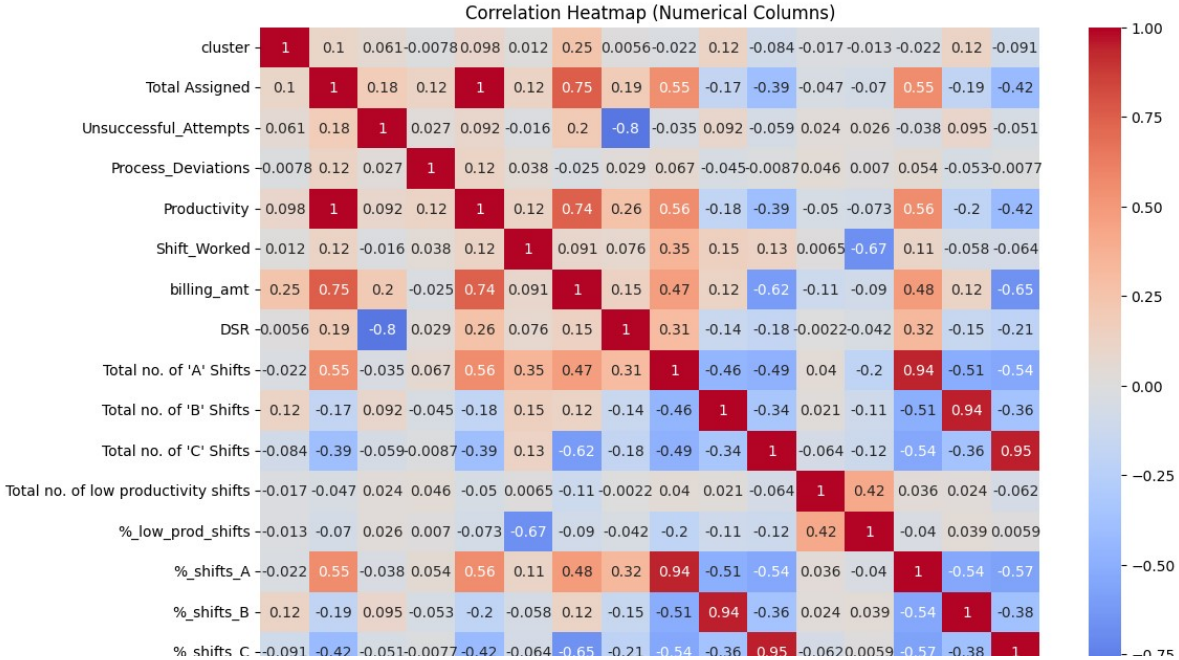
In [107]:
```
numeric_col=final_df.select_dtypes(include=['number']).columns
binary_categorical_columns = ['DSR Category', 'Regularity Classification']
```

In [110]:
```
correlation_matrix_num = final_df[numeric_col].corr()
correlation_matrix_num
```

Out[110]:

	cluster	Total Assigned	Unsuccessful_Attempts	Process_Deviations	Produc
cluster	1.000000	0.102152	0.060776	-0.007817	0.0
Total Assigned	0.102152	1.000000	0.181314	0.124430	0.9
Unsuccessful_Attempts	0.060776	0.181314	1.000000	0.026933	0.0
Process_Deviations	-0.007817	0.124430	0.026933	1.000000	0.1
Productivity	0.097887	0.995970	0.092383	0.123530	1.0
Shift_Worked	0.011837	0.117526	-0.016333	0.038482	0.1
billing_amt	0.245928	0.748901	0.199346	-0.025480	0.7
DSR	0.005644	0.189348	-0.800750	0.029165	0.2
Total no. of 'A' Shifts	-0.021988	0.547977	-0.035451	0.067309	0.5
Total no. of 'B' Shifts	0.120523	-0.174073	0.091607	-0.044810	-0.1
Total no. of 'C' Shifts	-0.084046	-0.394755	-0.059400	-0.008730	-0.3
Total no. of low productivity shifts	-0.017010	-0.047475	0.024330	0.045940	-0.0
%_low_prod_shifts	-0.013239	-0.070178	0.025509	0.007019	-0.0
%_shifts_A	-0.021705	0.552324	-0.037671	0.053907	0.5
%_shifts_B	0.117254	-0.191798	0.094595	-0.052992	-0.2
%_shifts_C	-0.090741	-0.422298	-0.050917	-0.007740	-0.4

In [111]:
```
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix_num, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Heatmap (Numerical Columns)')
plt.show()
```



Observation

1. After reading out the heatmap we can say that Total Assigned variable is highly correlated with productivity because the ration of unsuccessfull attempt was not included in productivity. Although we can say that the ratio of Unsuccessfull attempt was smaller
2. We can also see that if shift_worked was moving towards the positive side than % of low productivity shift correlation was moving to the negative side which mean if a person worked higher number of shift than there is higher number of chnace that they give lower productivity shift in between of their worked.
3. There is a positive correlation between DSR (Delivery Success rate) and Total Assigned variable which mean if we want to increase the DSR then we need to increased the total assigned value as well with each Delivery agent.
4. There is a positive correlation between DSR and billing amount as well . If we increase the DSR then we will be able to generate more revenue which refer billing amount here.
5. There is a strong negative correlation between Process deviation and Billing amount which mean we need to decrease the deviation of delivery agent if we want to increase our revenue in their shift
6. There is a negative correlation between DSR and %_shift_c which mean the success rate of delivery was less in night shift and higher in shift A

Method

We have choosen heatmap correlation method because in heatmap we can clearly see the negative and positive correlation

Part-B

Explain what you have gathered and understood about this data after having examined the correlations.

Data Gathered

1. After reading out the heatmap we can say that Total Assigned variable is highly correlated with productivity because the ration of unsuccessfull attempt was not included in productivity. Although we can say that the ratio of Unsuccessfull attempt was smaller
2. We can also see that if shift_worked was moving towards the positive side than % of low productivity shift correlation was moving to the negative side which mean if a person worked higher number of shift than there is higher number of chnace that they give lower productivity shift in between of their worked.
3. There is a positive correlation between DSR (Delivery Success rate) and Total Assigned variable which mean if we want to increase the DSR then we need to increased the total assigned value as well with each Delivery agent.
4. There is a positive correlation between DSR and billing amount as well . If we increase the DSR then we will be able to generate more revenue which refer billing amount here.
5. There is a strong negative correlation between Process deviation and Billing amount which mean we need to decrease the deviation of delivery agent if we want to increase our revenue in their shift
6. There is a negative correlation between DSR and %_shift_c which mean the success rate of delivery was less in night shift and higher in shift A

Question 7

Following a comprehensive analysis of the final_df, the next phase involves categorizing delivery agents into distinct clusters. The primary aim of establishing these clusters is to uncover specific and noteworthy attributes within each group, enabling the formulation of targeted strategies for improvement.

In this scenario, the objective is to segment delivery agents into various groups, facilitating an understanding of which group requires enhancements in specific metrics. Typically, exceptional performing delivery agents exhibit elevated DSR (Delivery Success Rate), minimal deviations, heightened productivity, and consistent work attendance. Conversely, underperforming agents display contrasting characteristics. Furthermore, there will be individuals who fall within intermediate ranges.

During the creation of agent groups, it's essential to facilitate the breakdown into smaller subsets within the population. This breakdown aids in pinpointing precise factors contributing to subpar performance.

Part-A

- Keeping the context in mind, choose and implement a method to perform this grouping. Create a new column to indicate which group each delivery agent belongs to. Explain why you have chosen this method.
- How many groups have you chosen to create and why?

IMPORTANT NOTES:

- You don't necessarily have to use every single feature for the grouping. You may choose to leave out some variables and explain why you have chosen to leave them out.
- Remember that the final_df contains several features which were created from original features in the raw data so that analysis and grouping becomes easier. Carefully go through all the features we currently have and discard what you think is not required when grouping on the basis of performance.
- Remember that performances are comparable across geographical areas. There is no reason to have different standards in different cities.
- You may choose to use hard coded rules for grouping the delivery agents (OR) you may even choose to use ML algorithms. Both approaches are valid as long as you are able to justify the logic and reasoning behind your approach.
- If you're finding it difficult to decide how many groups to divide them into, don't worry, this is common when dealing with data in the real world. Finding an appropriate solution requires trying out multiple approaches and analyzing results to select the method that seems to make most sense or fetches the highest scores on certain metrics.

In [171]:
```
''' Here we have taken four column[DSR,Process_Deviation, Productivity, Total
so that we can calculate the performance group of the delivery agent in respec
final_df['Performance Group'] = ''

for index, row in final_df.iterrows():
    if row["DSR"] >= 0.9 and row["Process_Deviations"] <= 2 and row["Productiv
    and row["Total no. of low productivity shifts"] <= 2 and row["Regularity C
        final_df.at[index, 'Performance Group'] = 'High'
    elif row["DSR"] >= 0.7 and row["Deviation Category"] == "deviations-ok" an
    and row["Total no. of low productivity shifts"] <= 1 and row["Regularity C
        final_df.at[index, 'Performance Group'] = 'Intermediate'
    else:
        final_df.at[index, 'Performance Group'] = 'Low'
```

Explanation

In this approach, we've defined specific rules based on the provided description for categorizing delivery agents into "High," "Intermediate," and "Low" performers. We loop through each row in the dataframe and apply these rules. If the conditions for a particular group are met, we assign that group label to the respective delivery agent.

In [ ]:
```
''' For calculation part we have put it down those column in a dataframe and s
the range of percentile for above column . Hence we have put it down the value
```

In [204]:
```
final_df[final_df["Performance Group"]=="High"].shape
# We found  6 row which is basically for one delivery agent id hence only one
final_df[final_df["Performance Group"]=="Intermediate"].drop_duplicates().shap
# We found  9953 row which is basically for 9626 delivery agent id hence there
final_df[final_df["Performance Group"]=="Low"].drop_duplicates().shape
```

Out[204]:
```
(9626, 25)
```

Part-B

Now that the delivery agents are grouped, explore the descriptive statistics or distributions across all the features of each group. Based on that, make your recommendations for what areas need to be improved for each group.

If you'd like to, you may use the same methods used in previous questions to explore distributions.

Recommendation

In order to enhance the performance of our delivery agent partners, a high level of focus is required. The primary goal is to effectively reduce the Delivery Success Rate (DSR) in order to enhance the overall operational efficiency of our delivery processes.

To achieve this objective, one of the key strategies we can implement is to provide our delivery agents with a well-structured and well-thought-out skill enhancement program. This program should include targeted training sessions and skill-building exercises that address the specific areas where performance improvements are needed.

A crucial aspect of this strategy is to introduce strategically timed break periods during the course of the delivery agents' shifts. These breaks are designed to serve as periods of rest and recuperation, allowing the agents to recharge and refocus. This tactic not only prevents burnout and fatigue but also contributes to a reduction in the overall stress levels experienced by the agents. As a positive outcome, we can expect to witness a noticeable decrease in the instances of low productivity during the agents' shifts.

By giving our delivery agents the opportunity to take essential breaks, we are fostering an environment where their performance can be optimized. The scheduled breaks act as an effective means to prevent instances of prolonged inefficiency and downtime, leading to a higher DSR and improved overall delivery success.

2. Targeted Approach for Low-Level Workers:

Our commitment to achieving superior delivery performance extends to workers who currently fall within the lower performance category. It is imperative to recognize the potential within these individuals and provide them with the necessary support to facilitate their growth and

advancement.

In this regard, our strategy entails a two-pronged approach:

Firstly, we need to invest in comprehensive training programs that cater specifically to the needs of workers in the low-level category. These training sessions should be carefully curated to address the specific skill gaps and challenges that have contributed to their current performance status. By focusing on these areas, we can empower these workers to acquire the skills and knowledge required to elevate their performance to at least the intermediate level.

Additionally, we must actively organize and conduct workshops and skill development sessions on a regular basis. These workshops should not only provide valuable insights but also serve as platforms for interactive learning and engagement. Recognizing the achievements and progress of these workers through a structured recognition program is essential to boost their morale and motivation.

Ultimately, our objective is to enable these workers to progress from their current low-level status to the intermediate level category. By extending consistent support, training, and recognition, we can unlock their potential and contribute to a substantial improvement in their performance. This approach aligns with our commitment to cultivating a diverse and skilled workforce that collectively contributes to the enhanced success of our delivery operations.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: