E-commerce Suply Chain Analysis

Python(pandas , numpy) , SQL , Power bi

Prince kumar yadav

```python
import pandas as pd
import numpy as np

# Load files

orders = pd.read_csv('order_details_table.csv')
delivery_person = pd.read_csv('delivery_person_table.csv')

# Standardize column names

orders.columns = orders.columns.str.strip().str.lower().str.replace(' ','_')
delivery_person.columns = delivery_person.columns.str.strip().str.lower().str.replace(' ','_')

print(orders.describe())
print(delivery_person.describe())

#  Handle missing values In Tables

print('orders-\n' ,orders.isnull().sum())
print('delivery_person-\n' ,delivery_person.isnull().sum())

# Fill missing ratings with mean

delivery_person['delivery_person_ratings'] = pd.to_numeric(delivery_person[
    'delivery_person_ratings'] , errors= 'coerce')

delivery_person['delivery_person_ratings'] = delivery_person['delivery_person_ratings'].fillna(
    delivery_person['delivery_person_ratings'].mean())
```

```python

# Fill missing ages with median

delivery_person['delivery_person_age']= pd.to_numeric(delivery_person['delivery_person_age'], errors= 'coerce')

delivery_person['delivery_person_age']= delivery_person['delivery_person_age'].fillna(
    delivery_person['delivery_person_age'].median())


# Handle Duplicates

print('orders-\n' ,orders.duplicated().sum())
print('delivery_person-\n' ,delivery_person.duplicated().sum())

# No Duplicates

# Clean Time_takenmin column

orders['time_taken(min)'] = orders['time_taken(min)'].str.replace(r'\(min\)\s*', '', regex=True)

# # Convert to numeric

orders['time_taken(min)'] = pd.to_numeric(orders['time_taken(min)'], errors='coerce')

print(orders['time_taken(min)'].dtype)

orders['order_date'] = pd.to_datetime(orders['order_date'], format='%d-%m-%Y')


# to save

orders.to_csv("cleaned_orders.csv", index=False)
delivery_person.to_csv("cleaned_delivery_person.csv", index=False)
```

Python to SQL Data transfer

```python
from sqlalchemy import create_engine

# Replace with your database details
user = "root"
password = "****"
host = "localhost"
database = "supply_chain_analysis"

# Create connection engine
engine = create_engine(f"mysql+pymysql://{'root'}:{'*****'}@{'localhost'}/{'supply_chain_analysis'}")

# Save orders table to SQL
orders.to_sql("orders", engine, index=False, if_exists="replace")


# Save delivery_person table
delivery_person.to_sql("delivery_person", engine, index=False, if_exists="replace")

print("✅ Data successfully uploaded to SQL database!")
```

```sql
create database supply_chain_analysis;
use supply_chain_analysis;

select count(*) from delivery_person ;
select count(*) from location ;
select count(*) from orders;

----- Total Orders Placed -----

SELECT
    COUNT(*) AS total_orders
FROM
    orders;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_orders
45593

```sql
-- Orders by City --

SELECT
    city , COUNT(id) AS total_orders
FROM
    orders
GROUP BY city
ORDER BY total_orders desc;
```

Result Grid | Filter Rows: | Export: | Wrap

city	total_orders
Metropolitian	34093
Urban	10136
NaN	1200
Semi-Urban	164

```sql
-- Rename column --

ALTER TABLE orders
CHANGE `time_taken(min)` time_taken_min BIGINT;

-- Average Delivery Time by City --

SELECT
    city, ROUND(AVG(time_taken_min), 2) AS avg_delivery_time
FROM
    orders
GROUP BY city
ORDER BY avg_delivery_time DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

city	avg_delivery_time
Semi-Urban	49.73
Metropolitian	27.32
Urban	22.98
NaN	22.06

```sql
-- Average Delivery Time by Weather --

SELECT
    weatherconditions,
    ROUND(AVG(time_taken_min), 2) AS avg_delivery_time
FROM
    orders
GROUP BY weatherconditions
ORDER BY avg_delivery_time DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

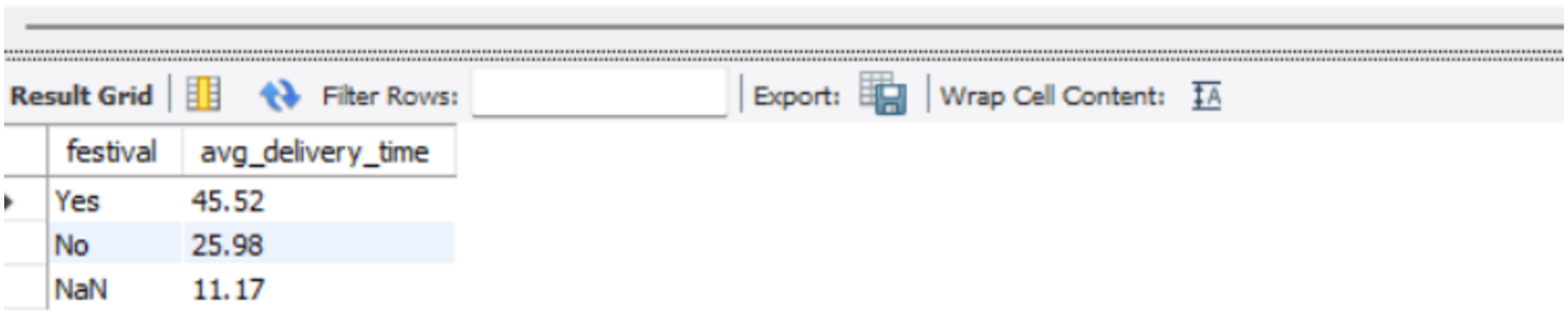
weatherconditions	avg_delivery_time
conditions Cloudy	28.92
conditions Fog	28.92
conditions NaN	26.55
conditions Windy	26.12
conditions Sandstorms	25.88
conditions Stormy	25.87
conditions Sunny	21.86

```sql
-- Impact of Traffic on Delivery Time --

SELECT
    road_traffic_density,
    ROUND(AVG(time_taken_min), 2) AS avg_delivery_time
FROM
    orders
GROUP BY 1
ORDER BY 2 DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

road_traffic_density	avg_delivery_time
Jam	31.18
High	27.24
Medium	26.70
NaN	26.54
Low	21.27

```sql
-- Festival Impact on Delivery Time --

SELECT
    festival, ROUND(AVG(time_taken_min), 2) AS avg_delivery_time
FROM
    orders
GROUP BY 1
ORDER BY 2 DESC;
```



festival	avg_delivery_time
Yes	45.52
No	25.98
NaN	11.17

```sql
-- Top 5 Fastest Delivery Persons --

SELECT
    delivery_person_id,
    ROUND(AVG(time_taken_min), 2) AS avg_delivery_time
FROM
    delivery_person
        JOIN
    orders ON delivery_person.id = orders.id
GROUP BY 1
ORDER BY 2 asc limit 5 ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch ro

delivery_person_id	avg_delivery_time
KOLRES06DEL03	19.44
DEHRES20DEL03	19.60
DEHRES17DEL01	19.75
KOLRES01DEL03	19.82
KNPRES01DEL01	20.00

```sql
-- Average Delivery Person Age & Rating by City --

SELECT
    city,
    ROUND(AVG(delivery_person_age)) AS avg_age,
    ROUND(AVG(delivery_person_ratings), 2) AS avg_rating
FROM
    delivery_person
        JOIN
    orders ON delivery_person.id = orders.id
GROUP BY 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

city	avg_age	avg_rating
Urban	29	4.67
Metropolitian	30	4.62
Semi-Urban	32	4.5
NaN	29	4.67

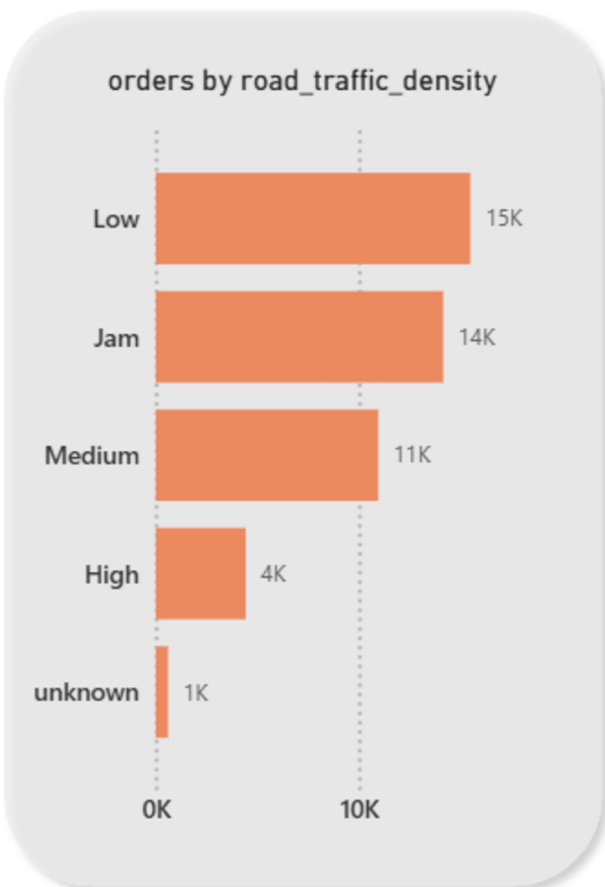
```
99      -- Orders per Type of Order --
100
101 •   SELECT
102         type_of_order, COUNT(id) AS orders
103     FROM
104         orders
105     GROUP BY 1;
```

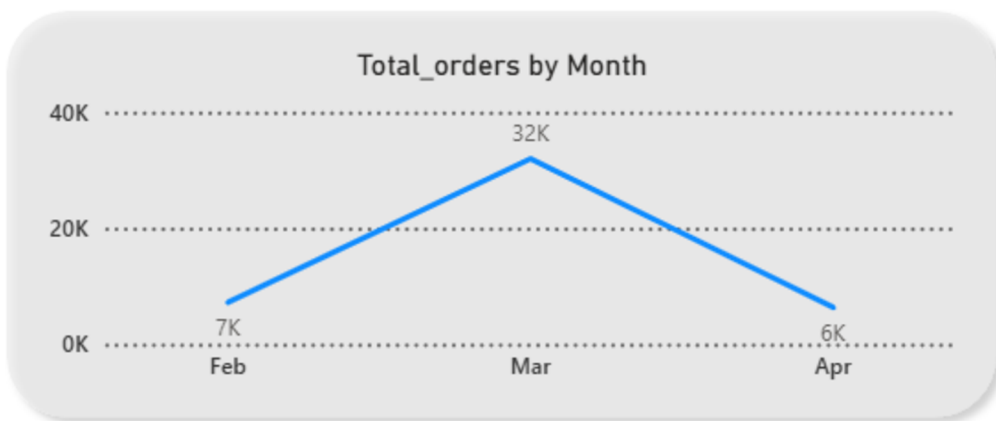
Result Grid | Filter Rows: | Export: | Wrap C

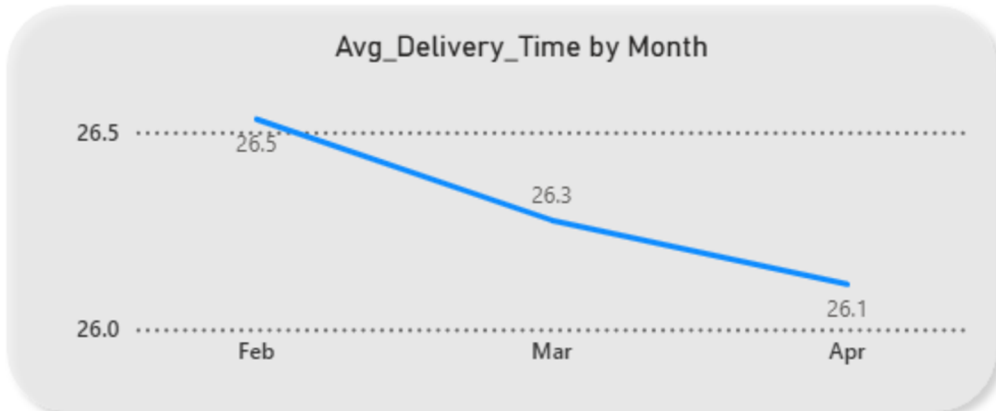
type_of_order	orders
Snack	11533
Drinks	11322
Buffet	11280
Meal	11458

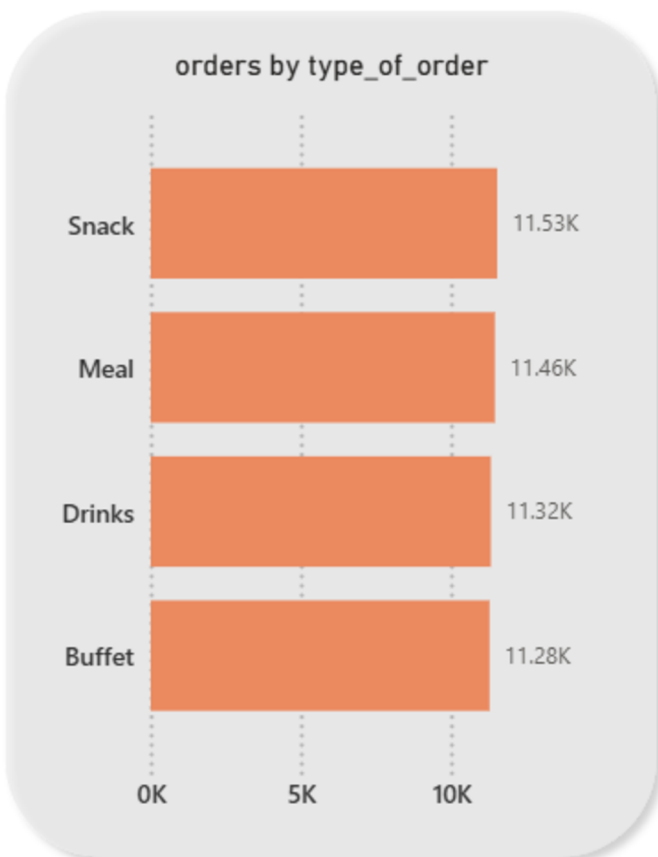
E-Commerce Supply Chain Analysis Dashboard

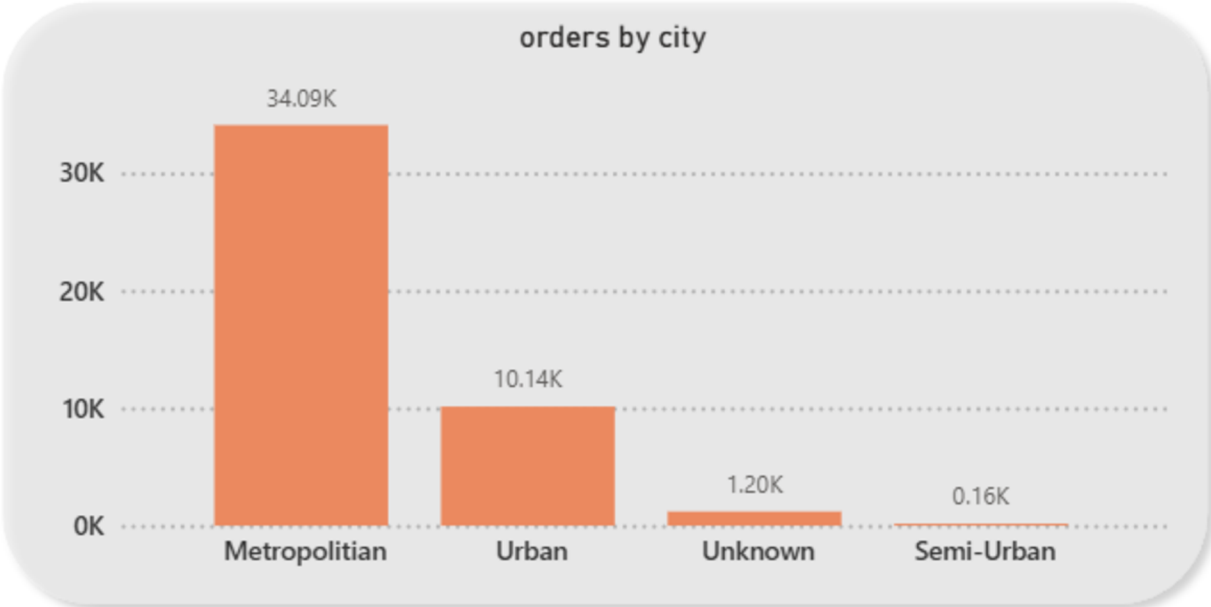
Total_orders	Avg_Delivery_Time	Avg_Driver_Age	Avg_Driver_Rating	Delivery_Person
45.59K	26.3	30	4.63	1320

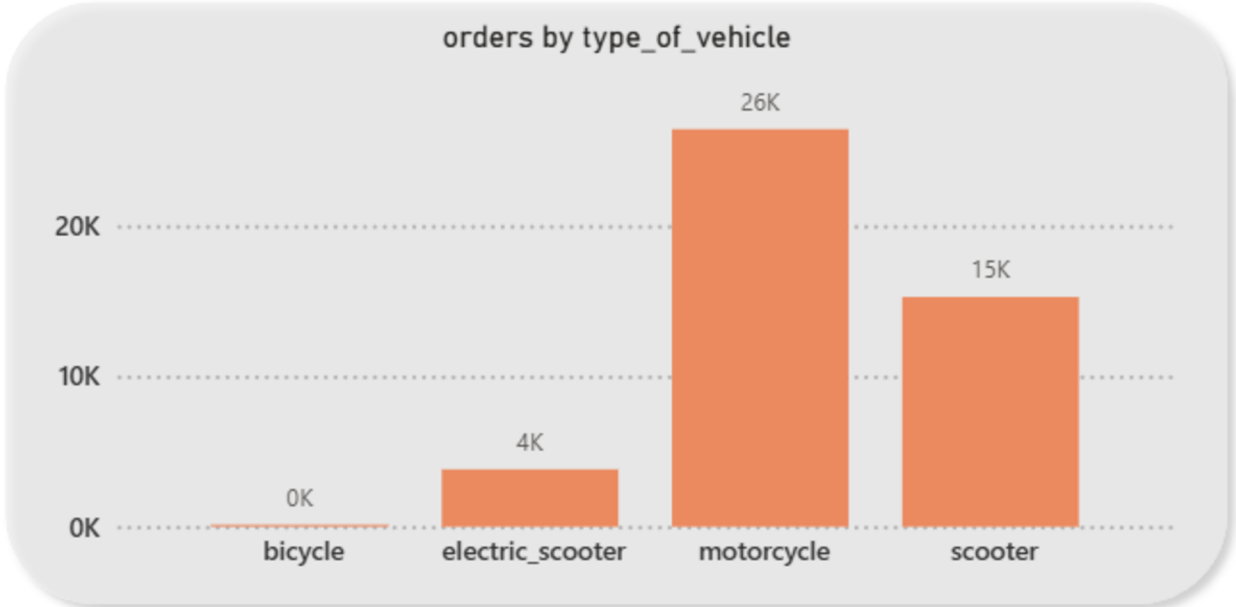


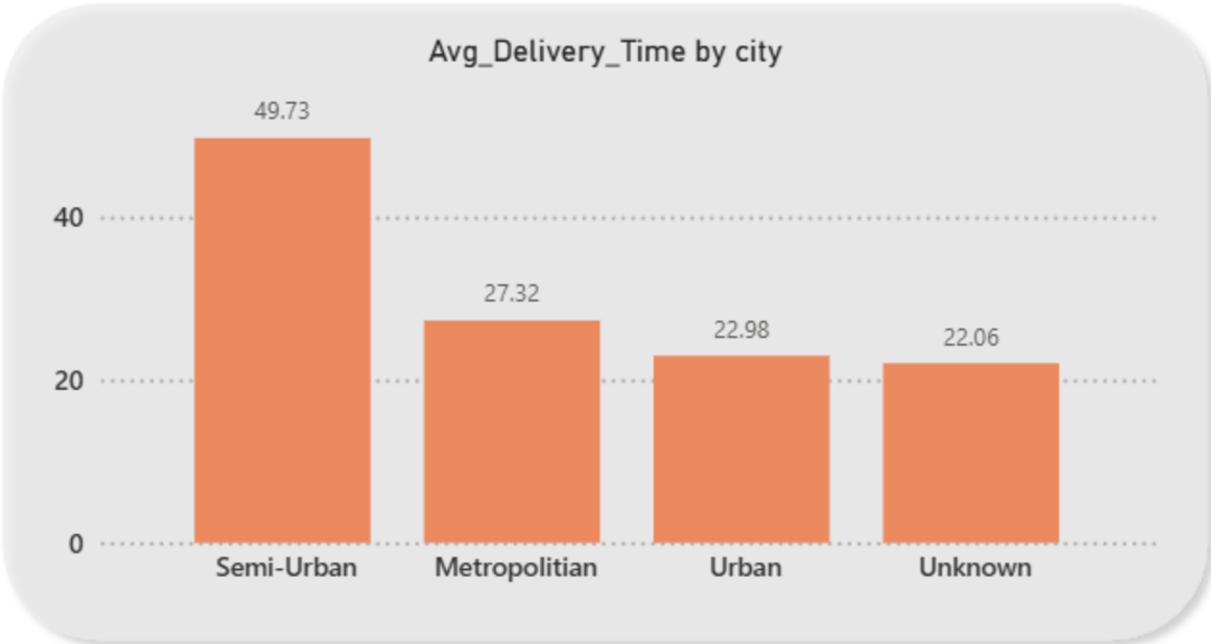


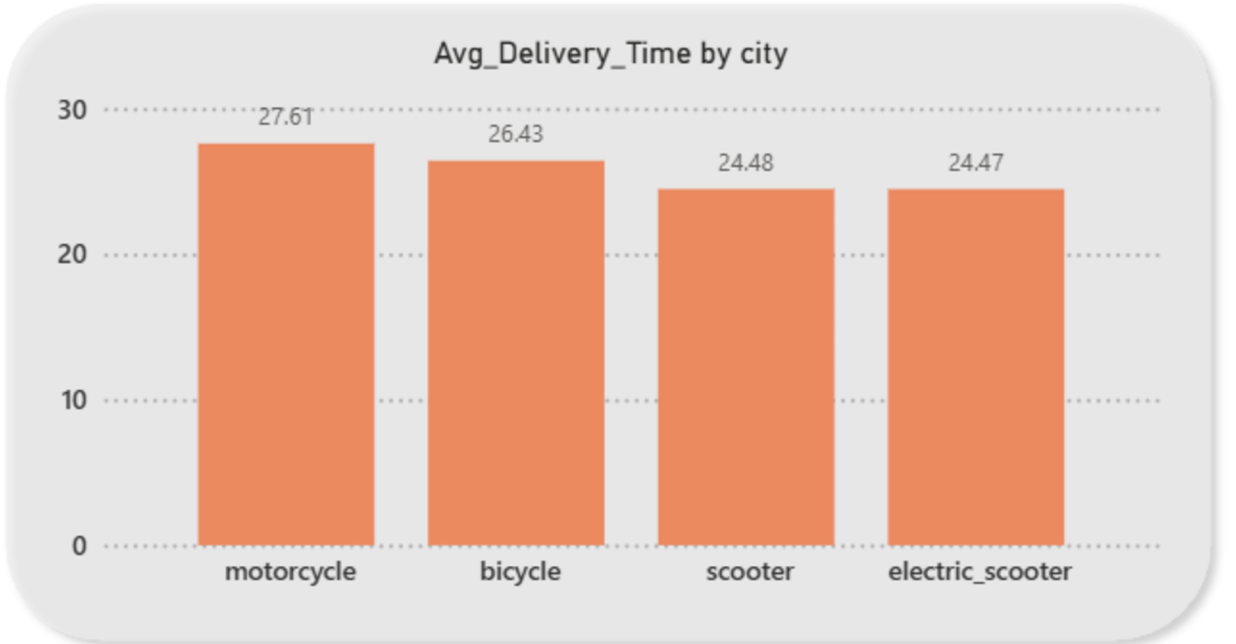


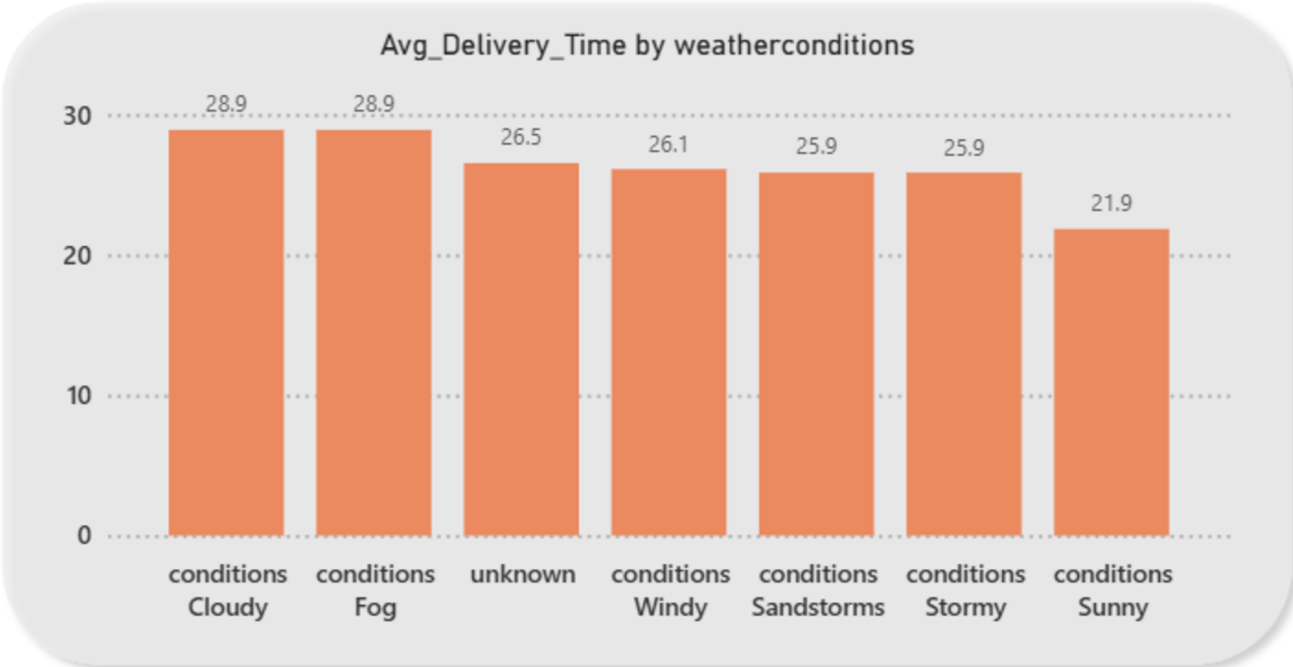


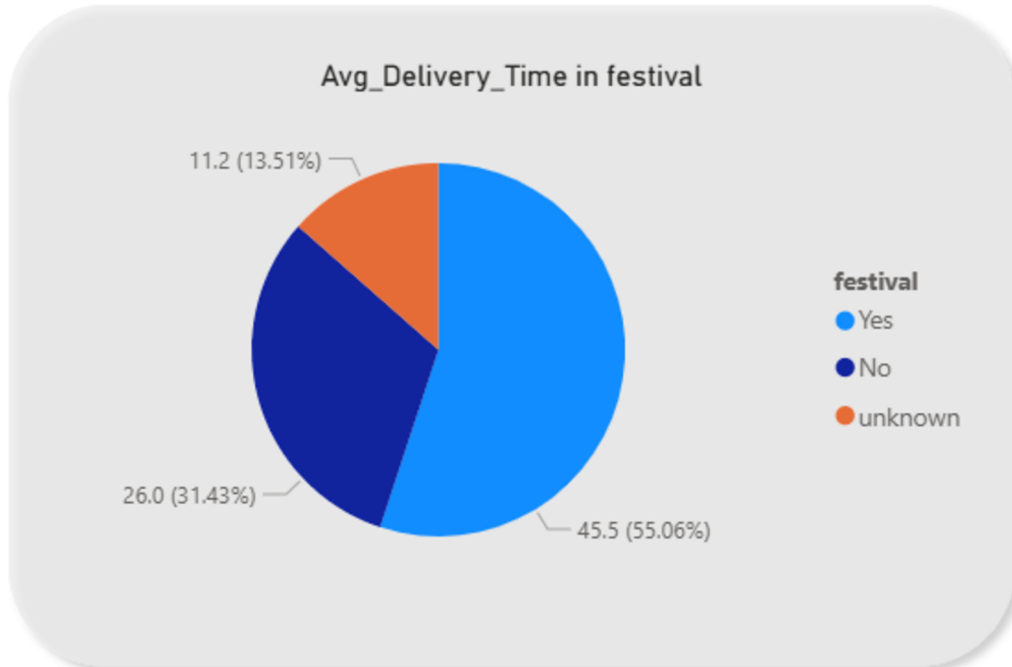

orders by city
34.09K
30K
20K
10.14K
10K
1.20K
0.16K
0K
Metropolitian
Urban
Unknown
Semi-Urban


orders by type_of_vehicle
26K
20K
15K
10K
4K
0K
0K
bicycle
electric_scooter
motorcycle
scooter


Avg_Delivery_Time by city
49.73
40
27.32
22.98
22.06
20
0
Semi-Urban
Metropolitian
Urban
Unknown


Avg_Delivery_Time by city
30
27.61
26.43
24.48
24.47
20
10
0
motorcycle
bicycle
scooter
electric_scooter





Slicers

city
- Metropolitian
- Semi-Urban
- Unknown
- Urban

Monthname
- Feb
- Mar
- Apr

road_traffic_density
- High
- Medium
- Jam
- unknown
- Low

type_of_vehicle
- bicycle
- electric_scooter
- motorcycle
- scooter