```
import pandas as pd

# ------------------------
# 1. Raw Vehicle Counts
# ------------------------
nsp_to_pitampura = pd.DataFrame({
    "Time": ["3:30-4:30", "4:30-5:30", "5:30-6:30", "6:30-7:30"],
    "Bus/Truck": [72, 68, 56, 40],
    "Car+Van": [1304, 1160, 1068, 888],
    "Motorcycle": [1068, 992, 728, 716],
    "Autorickshaw": [0, 172, 144, 168],
    "Total": [2664, 2392, 1996, 1812]
})

pitampura_to_nsp = pd.DataFrame({
    "Time": ["3:30-4:30", "4:30-5:30", "5:30-6:30", "6:30-7:30"],
    "Bus/Truck": [60, 64, 36, 24],
    "Car+Van": [1200, 1244, 1036, 840],
    "Motorcycle": [936, 792, 532, 600],
    "Autorickshaw": [140, 100, 80, 144],
    "Total": [2336, 2200, 1684, 1608]
})

# ------------------------
# 2. Vehicle Composition (%)
# ------------------------
vehicle_composition = pd.DataFrame({
    "Direction": ["NSP→Pitampura", "Pitampura→NSP"],
    "Bus/Truck (%)": [2.66, 2.25],
    "Car+Van (%)": [49.86, 55.24],
    "Motorcycle (%)": [39.53, 36.57],
    "Autorickshaw (%)": [7.94, 5.93]
})

# ------------------------
# 3. PCU Factors
# ------------------------
pcu_factors = pd.DataFrame({
    "Vehicle": ["Bus/Truck", "Car+Van", "Motorcycle", "Auto-rickshaw"],
    "PCU Value": [2.2, 1, 0.5, 1.2]
})

# ------------------------
# 4. Flow Rates (PCU/hr)
# ------------------------
flow_rates_nsp_to_pit = pd.DataFrame({
    "Time": ["3:30-4:30", "4:30-5:30", "5:30-6:30", "6:30-7:30"],
    "Flow Rate (PCU/hr)": [2260.4, 2012, 1728, 1536]
})

flow_rates_pit_to_nsp = pd.DataFrame({
    "Time": ["3:30-4:30", "4:30-5:30", "5:30-6:30", "6:30-7:30"],
    "Flow Rate (PCU/hr)": [1968, 1900.8, 1477.2, 1348]
})

# ------------------------
# 5. Directional Distribution
# ------------------------
directional_distribution = pd.DataFrame({
    "Direction": ["NSP→Pitampura", "Pitampura→NSP"],
    "PCU/hr": [2260.4, 1968.0],
    "Total": [4228.4, 4228.4],
    "Distribution (%)": [53.45, 46.54]
})

# ------------------------
# 6. ADT and AADT
# ------------------------
adt_aadt = pd.DataFrame({
    "Direction": ["NSP→Pitampura", "Pitampura→NSP"],
    "ADT (PCU)": [28195, 24972],
    "AADT (PCU)": [33440, 29617]
})

# ------------------------
# 7. Average Velocity (Greenshield)
# ------------------------
avg_velocity = pd.DataFrame({
    "Direction": ["NSP→Pitampura", "Pitampura→NSP"],
    "Average Velocity (km/hr)": [8.574, 9.248]
})
```

```
# ------------------------
# Export to Excel
# ------------------------
with pd.ExcelWriter("Traffic_Volume_Study.xlsx", engine="openpyxl") as writer:
    nsp_to_pitampura.to_excel(writer, sheet_name="Raw Counts NSP→Pit", index=False)
    pitampura_to_nsp.to_excel(writer, sheet_name="Raw Counts Pit→NSP", index=False)
    vehicle_composition.to_excel(writer, sheet_name="Vehicle Composition", index=False)
    pcu_factors.to_excel(writer, sheet_name="PCU Factors", index=False)
    flow_rates_nsp_to_pit.to_excel(writer, sheet_name="Flow NSP→Pit", index=False)
    flow_rates_pit_to_nsp.to_excel(writer, sheet_name="Flow Pit→NSP", index=False)
    directional_distribution.to_excel(writer, sheet_name="Directional Distribution", index=False)
    adt_aadt.to_excel(writer, sheet_name="ADT & AADT", index=False)
    avg_velocity.to_excel(writer, sheet_name="Average Velocity", index=False)

print("✅ Excel file 'Traffic_Volume_Study.xlsx' created successfully.")
```

```
✅ Excel file 'Traffic_Volume_Study.xlsx' created successfully.
```

```
import pandas as pd

# File path (update this with your actual path)
file_path = "Traffic_Volume_Study.xlsx"

# Define PCU factors
pcu_factors = {
    "Bus/Truck": 2.2,
    "Car+Van": 1.0,
    "Motorcycle": 0.5,
    "Autorickshaw": 1.2
}

# Function to calculate PCU for each row
def calculate_pcu(row):
    return (row.get("Bus/Truck", 0) * pcu_factors["Bus/Truck"] +
            row.get("Car+Van", 0) * pcu_factors["Car+Van"] +
            row.get("Motorcycle", 0) * pcu_factors["Motorcycle"] +
            row.get("Autorickshaw", 0) * pcu_factors["Autorickshaw"])

# Load the Excel file
xls = pd.ExcelFile(file_path)

# Dictionary to hold processed data
processed_data = {}

# Loop through all sheets
for sheet in xls.sheet_names:
    df = pd.read_excel(file_path, sheet_name=sheet)

    # Check if required columns exist in the sheet
    required_cols = {"Bus/Truck", "Car+Van", "Motorcycle", "Autorickshaw"}
    if required_cols.issubset(df.columns):
        df["PCU/hr"] = df.apply(calculate_pcu, axis=1)

    processed_data[sheet] = df

# Save results into a new Excel file
output_path = "Traffic_Volume_with_PCU_AllSheets.xlsx"
with pd.ExcelWriter(output_path, engine="openpyxl") as writer:
    for sheet_name, df in processed_data.items():
        df.to_excel(writer, sheet_name=sheet_name, index=False)

print("File saved as:", output_path)
```

```
File saved as: Traffic_Volume_with_PCU_AllSheets.xlsx
```

```
import pandas as pd

# File path (update this with your file location)
file_path = "Traffic_Volume_with_PCU_AllSheets.xlsx"

# Load the processed sheets
df_nsp_pit = pd.read_excel(file_path, sheet_name="Raw Counts NSP→Pit")
df_pit_nsp = pd.read_excel(file_path, sheet_name="Raw Counts Pit→NSP")

# Find peak hour for each direction
peak_nsp_pit = df_nsp_pit.loc[df_nsp_pit["PCU/hr"].idxmax()]
peak_pit_nsp = df_pit_nsp.loc[df_pit_nsp["PCU/hr"].idxmax()]

print("NSP → Pitampura Peak Hour:")
```

```
print(f"Time: {peak_nsp_pit['Time']}, PCU/hr: {peak_nsp_pit['PCU/hr']}")

print("\nPitampura → NSP Peak Hour:")
print(f"Time: {peak_pit_nsp['Time']}, PCU/hr: {peak_pit_nsp['PCU/hr']}")
```

```
NSP → Pitampura Peak Hour:
Time: 4:30-5:30, PCU/hr: 2012.0

Pitampura → NSP Peak Hour:
Time: 3:30-4:30, PCU/hr: 1968.0
```

```
import pandas as pd

# File path (update this with your file location)
file_path = "Traffic_Volume_with_PCU_AllSheets.xlsx"

# Load the processed sheets
df_nsp_pit = pd.read_excel(file_path, sheet_name="Raw Counts NSP→Pit")
df_pit_nsp = pd.read_excel(file_path, sheet_name="Raw Counts Pit→NSP")

# Find peak hour for each direction
peak_nsp_pit = df_nsp_pit.loc[df_nsp_pit["PCU/hr"].idxmax()]
peak_pit_nsp = df_pit_nsp.loc[df_pit_nsp["PCU/hr"].idxmax()]

# IRC standard for 2-lane undivided urban road
irc_capacity = 2400

def check_saturation(peak_value):
    if peak_value >= irc_capacity:
        return "Above capacity → Severe congestion"
    elif peak_value >= 0.9 * irc_capacity:
        return "Nearly saturated → High congestion"
    else:
        return "Below capacity → Traffic flow acceptable"

# Results
print("NSP → Pitampura")
print(f"Peak Hour: {peak_nsp_pit['Time']}, PCU/hr: {peak_nsp_pit['PCU/hr']}")
print("Status:", check_saturation(peak_nsp_pit["PCU/hr"]))

print("\nPitampura → NSP")
print(f"Peak Hour: {peak_pit_nsp['Time']}, PCU/hr: {peak_pit_nsp['PCU/hr']}")
print("Status:", check_saturation(peak_pit_nsp["PCU/hr"]))
```

```
NSP → Pitampura
Peak Hour: 4:30-5:30, PCU/hr: 2012.0
Status: Below capacity → Traffic flow acceptable

Pitampura → NSP
Peak Hour: 3:30-4:30, PCU/hr: 1968.0
Status: Below capacity → Traffic flow acceptable
```

```
import pandas as pd

# Peak PCU values (you can also fetch these from earlier calculations)
peak_nsp_pit = 2260.4
peak_pit_nsp = 1968.0

# Step 5: Directional Distribution
total_peak = peak_nsp_pit + peak_pit_nsp
dist_nsp_pit = (peak_nsp_pit / total_peak) * 100
dist_pit_nsp = (peak_pit_nsp / total_peak) * 100

print("Step 5: Directional Distribution")
print(f"NSP → Pitampura: {dist_nsp_pit:.2f}%")
print(f"Pitampura → NSP: {dist_pit_nsp:.2f}%")

if dist_nsp_pit > dist_pit_nsp:
    print("Interpretation: Evening peak direction dominance (NSP → Pitampura).")
else:
    print("Interpretation: Evening peak direction dominance (Pitampura → NSP).")

# Step 6: Level of Service (LOS)
# Based on the study → LOS = F (jammed flow, very poor quality of service)

print("\nStep 6: Level of Service (LOS)")
print("According to Greenshields' model and the paper's analysis:")
print("LOS = F → Jammed flow, oversaturated, very poor service quality.")
```

```
Step 5: Directional Distribution
NSP → Pitampura: 53.46%
Pitampura → NSP: 46.54%
```

```
Interpretation: Evening peak direction dominance (NSP → Pitampura).

Step 6: Level of Service (LOS)
According to Greenshields' model and the paper's analysis:
LOS = F → Jammed flow, oversaturated, very poor service quality.
```