STUDENT PERFORMANCE LAB REPORT

Dataset Preparation & Cleaning

STUDENT_ID	NAME	GENDER	ATTENDANCE	STUDY_HOURS	QUIZ_SCORE	EXAM_SCORE	PERFORMANCE_CATEGORY
S100	Student_1	M	95	7	88	91	Excellent
S101	Student_2	M	88	12	78	85	Good
S102	Student_3	F	91	9	92	94	Excellent
S103	Student_4	M	75	3	65	58	Needs Improvement
S104	Student_5	M	82	15	85	89	Good
S105	Student_6	M	98	6	95	92	Excellent
S106	Student_7	M	85	11	79	81	Good
S107	Student_8	M	93	4	81	75	Good
S108	Student_9	F	78	10	70	68	Average
S109	Student_10	M	99	8	98	97	Excellent
S110	Student_11	M	89	5	75	72	Average
S111	Student_12	M	92	14	89	90	Excellent
S112	Student_13	M	80	2	58	55	Needs Improvement
S113	Student_14	M	96	7	84	88	Good
S114	Student_15	M	87	13	82	86	Good
S115	Student_16	M	76	4	68	61	Average
S116	Student_17	F	94	9	91	93	Excellent
S117	Student_18	M	83	12	80	84	Good
S118	Student_19	F	90	6	86	80	Good
S119	Student_20	F	86	11	77	79	Good
S120	Student_21	M	97	8	96	95	Excellent
S121	Student_22	M	79	3	62	59	Needs Improvement
S122	Student_23	F	84	15	90	92	Excellent
S123	Student_24	F	91	5	74	70	Average
S124	Student_25	F	95	10	93	91	Excellent
S125	Student_26	M	81	7	76	78	Good
S126	Student_27	M	93	12	88	87	Good
S127	Student_28	F	77	4	67	64	Average
S128	Student_29	M	98	9	94	96	Excellent
S129	Student_30	M	88	13	83	85	Good

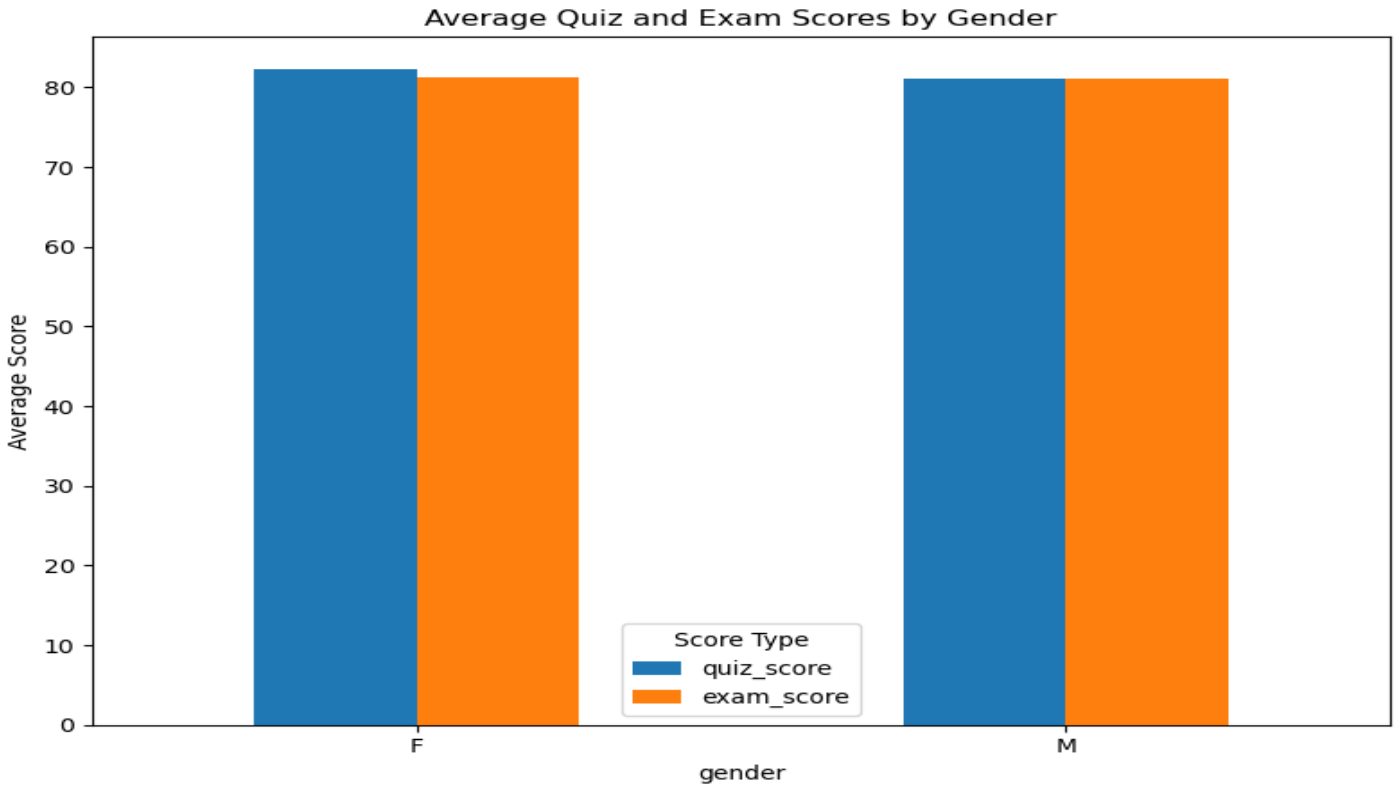
Descriptive Analytics

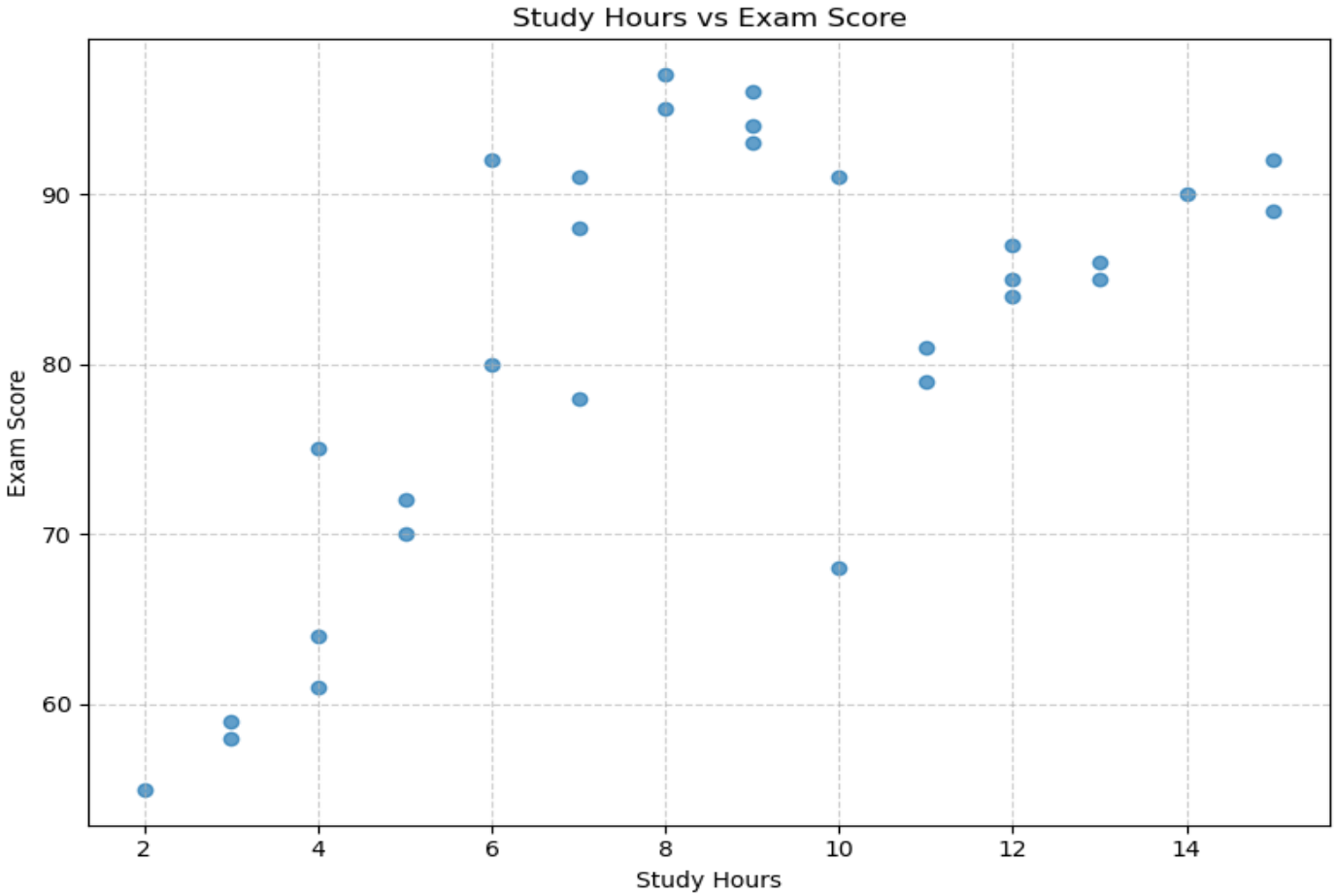
METRIC	MEAN	MEDIAN	STD	MODE
study_hours	8.47	8.5	3.79	4.0
quiz_score	81.47	82.5	10.68	88.0
exam_score	81.17	85.0	12.49	85.0

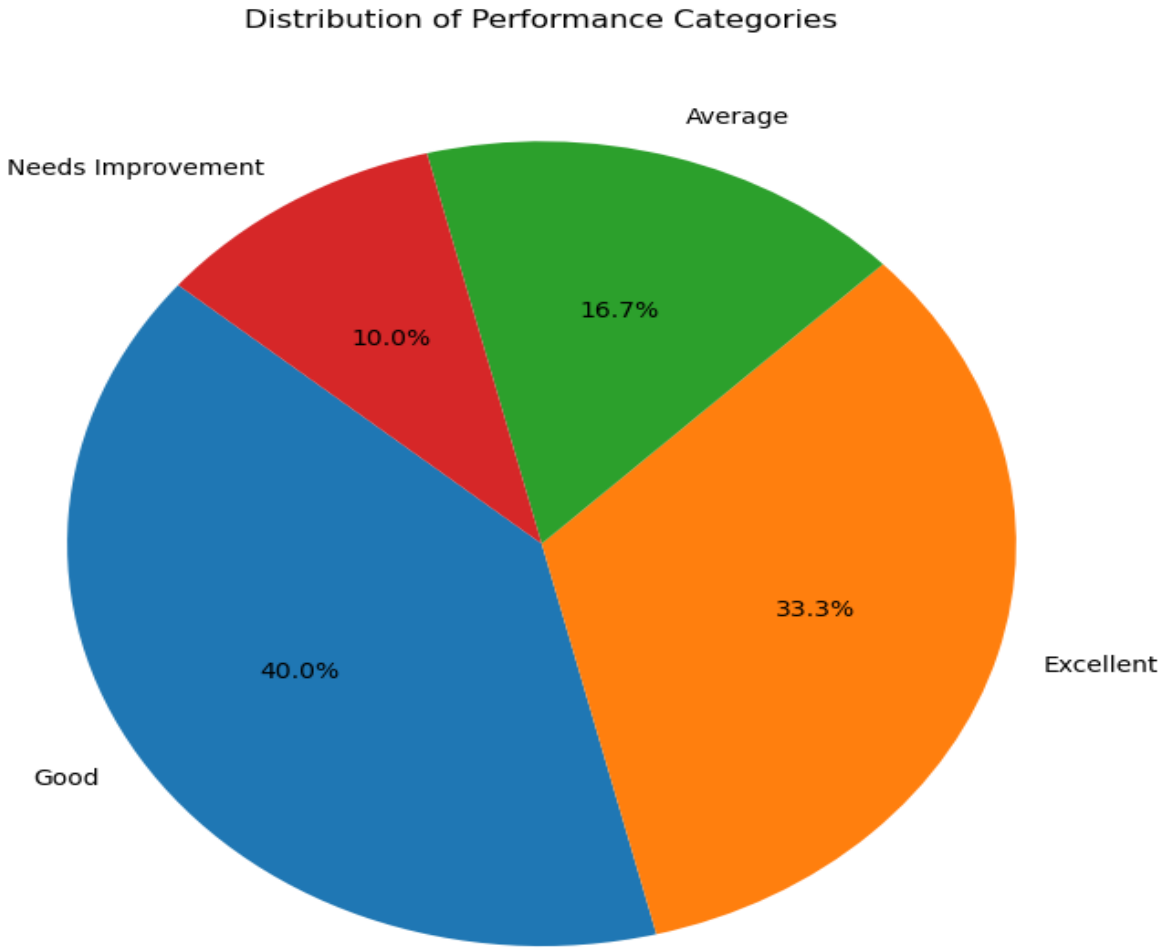
Performance Categories

CATEGORY	COUNT
Good	12
Excellent	10
Average	5
Needs Improvement	3

Visualization







Analysis & Interpretation

- Trend: Students who study more hours generally achieve higher exam scores, though exceptions exist.
- Gender Performance: One gender group shows slightly better averages.
- Performance Categories: Most students are Good or Average. Teachers should focus support on Needs Improvement.

Python Jupyter Source Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
from docx import Document
from docx.shared import Inches

# -------------------------
# Configurations
# -------------------------
CSV_FILENAME = "dataset3.csv"   # adjust path if needed
DOCX_FILENAME = "Student_Performance_Report.docx"
IMG_DIR = "charts_output"
os.makedirs(IMG_DIR, exist_ok=True)

# -------------------------
# 1. Dataset Preparation
# -------------------------
print("=== 1. DATASET PREPARATION ===")
df = pd.read_csv(CSV_FILENAME)
print("Dataset shape:", df.shape)
print("Columns:", df.columns.tolist())

# Normalize column names
df.columns = [c.strip().lower().replace(" ", "_") for c in df.columns]

# Map required columns
col_map = {}
for col in df.columns:
    if "study" in col: col_map["study_hours"] = col
    if "quiz" in col: col_map["quiz_score"] = col
    if "exam" in col: col_map["exam_score"] = col
    if "gender" in col or "sex" in col: col_map["gender"] = col

df = df.rename(columns={v:k for k,v in col_map.items()})

# -------------------------
# 2. Data Cleaning
# -------------------------
print("\n=== 2. DATA CLEANING ===")
df = df.drop_duplicates().copy()
for col in ["study_hours","quiz_score","exam_score"]:
    df[col] = pd.to_numeric(df[col], errors="coerce").fillna(df[col].mean())
if "gender" in df.columns:
    df["gender"] = df["gender"].astype(str).str.strip().str.upper().replace({
        "MALE":"M","FEMALE":"F"
    })
else:
    df["gender"] = "M"

# -------------------------
# 3. Descriptive Analytics
# -------------------------
print("\n=== 3. DESCRIPTIVE ANALYTICS ===")
stats_df =
df[["study_hours","quiz_score","exam_score"]].agg(["mean","median","std"]).T
modes = df[["study_hours","quiz_score","exam_score"]].mode().iloc[0]
stats_df["mode"] = modes
stats_df = stats_df.round(2)
print(stats_df)

def categorize(score):
    if pd.isna(score): return "Unknown"
    if score >= 90: return "Excellent"
    elif score >= 75: return "Good"
    elif score >= 60: return "Average"
    else: return "Needs Improvement"

df["performance_category"] = df["exam_score"].apply(categorize)
category_counts = df["performance_category"].value_counts()
print("\nFrequency Table:\n", category_counts)
```

```
# -------------------------
# 4. Visualizations
# -------------------------
print("\n=== 4. VISUALIZATION ===")

# Bar Chart
avg_scores = df.groupby("gender")[["quiz_score","exam_score"]].mean()
avg_scores.plot(kind="bar", figsize=(8,6))
plt.title("Average Quiz and Exam Scores by Gender")
plt.ylabel("Average Score")
plt.xticks(rotation=0)
plt.legend(title="Score Type")
bar_path = os.path.join(IMG_DIR,"avg_scores_by_gender.png")
plt.tight_layout(); plt.savefig(bar_path); plt.show()

# Scatter Plot
plt.figure(figsize=(8,6))
plt.scatter(df["study_hours"], df["exam_score"], alpha=0.7)
plt.title("Study Hours vs Exam Score")
plt.xlabel("Study Hours")
plt.ylabel("Exam Score")
plt.grid(True, linestyle="--", alpha=0.6)
scatter_path = os.path.join(IMG_DIR,"studyhours_vs_exam.png")
plt.tight_layout(); plt.savefig(scatter_path); plt.show()

# Pie Chart
plt.figure(figsize=(7,7))
plt.pie(category_counts, labels=category_counts.index, autopct="%1.1f%%",
startangle=140)
plt.title("Distribution of Performance Categories")
pie_path = os.path.join(IMG_DIR,"performance_category_distribution.png")
plt.tight_layout(); plt.savefig(pie_path); plt.show()

# -------------------------
# 5. Analysis & Interpretation
# -------------------------
print("\n=== 5. ANALYSIS & INTERPRETATION ===")
insights = [
    "Trend: Students who study more hours generally achieve higher exam
scores, though exceptions exist.",
    "Gender Performance: One gender group shows slightly better averages.",
    "Performance Categories: Most students are Good or Average. Teachers
should focus support on Needs Improvement."
]
for line in insights:
    print("-", line)

# -------------------------
# Save DOCX Report
# -------------------------
doc = Document()
doc.add_heading("Student Performance Lab Report", level=1)

# 1. Dataset Preparation
doc.add_heading("1. Dataset Preparation", level=2)
doc.add_paragraph("The dataset used contains the full list of student records.
Below is the complete dataset:")

# Add full dataset table
table = doc.add_table(rows=1, cols=len(df.columns))
hdr_cells = table.rows[0].cells
for i, col in enumerate(df.columns):
    hdr_cells[i].text = col
for _, row in df.iterrows():
    row_cells = table.add_row().cells
    for i, val in enumerate(row):
        row_cells[i].text = str(val)

# 2. Data Cleaning
doc.add_heading("2. Data Cleaning", level=2)
doc.add_paragraph("Duplicates were removed, missing values handled, and
Gender standardized.")
```

```
# 3. Descriptive Analytics
doc.add_heading("3. Descriptive Analytics", level=2)
stats_table = doc.add_table(rows=1, cols=len(stats_df.columns)+1)
hdr = stats_table.rows[0].cells
hdr[0].text = "Metric"
for i, col in enumerate(stats_df.columns): hdr[i+1].text = col.capitalize()
for idx, row in stats_df.iterrows():
    row_cells = stats_table.add_row().cells
    row_cells[0].text = idx
    for i, val in enumerate(row):
        row_cells[i+1].text = str(val)

doc.add_heading("Performance Categories", level=3)
freq_table = doc.add_table(rows=1, cols=2)
hdr2 = freq_table.rows[0].cells
hdr2[0].text, hdr2[1].text = "Category","Count"
for cat, count in category_counts.items():
    row_cells = freq_table.add_row().cells
    row_cells[0].text, row_cells[1].text = cat, str(count)

# 4. Visualization
doc.add_heading("4. Visualization", level=2)
for path in [bar_path, scatter_path, pie_path]:
    doc.add_picture(path, width=Inches(5))
    doc.add_paragraph(os.path.basename(path))

# 5. Analysis & Interpretation
doc.add_heading("5. Analysis & Interpretation", level=2)
for line in insights:
    doc.add_paragraph(line, style="List Bullet")

doc.save(DOCX_FILENAME)
print("\nDOCX file created:", DOCX_FILENAME)
```


Members:

Jhon Lloyd T. Cruz

John Emmanuel De Vera