Advanced Exam in Data Analytics, Statistical Methods, and Advanced Python Programming

Master 1, Data Science

Time allowed: 3 hours

Instructions: Answer all questions. Show all your work and clearly justify your reasoning. Use the provided data tables and charts for manual analysis where indicated.

1. **Advanced Concepts in Data Types and Measurement** (30 points)

   (a) (10 points) **Measurement Levels:**

   - Define and contrast the four levels of measurement (nominal, ordinal, interval, ratio).
   - Provide one real-world example for each.
   - Discuss how these levels impact the choice of statistical tests in analysis.

   (b) (10 points) **Standard Deviation and Z-Score Calculation:** Consider the following scenario: A quality control engineer collects the weights (in grams) of 10 products from a production line:

   $$490,\ 505,\ 500,\ 515,\ 495,\ 510,\ 505,\ 500,\ 520,\ 495.$$

   (a) Compute the mean and standard deviation (manually) of the product weights.

   (b) A product is found to weigh 530 grams. Calculate its Z-score and interpret its significance in the context of quality control.

   (c) (10 points) **Data Table Analysis:** Analyze the following table of socioeconomic data. Compute the mean, median, and mode for the **Income, Age** columns. Give Data Analytics explanation for Education Level and Socioeconomic Status. Then, discuss any potential issues (e.g., skewness, outliers) that might affect further statistical analysis.

ID	Age	Education Level	Income ($)	Socioeconomic Status
1	34	Bachelor	42000	Middle
2	45	Master	58000	Upper-Middle
3	29	Bachelor	35000	Lower-Middle
4	52	Ph.D.	75000	Upper
5	41	Master	50000	Middle
6	37	Bachelor	47000	Middle
7	28	Bachelor	33000	Lower-Middle
8	50	Ph.D.	81000	Upper

2. **Descriptive and Inferential Statistics with Z-Scores** (35 points)

   (a) (15 points) **Hypothesis Testing and Z-Score Application:** A study examines the weights of a large batch of products. Assume the weights are normally distributed with a mean of 500 grams and a standard deviation of 20 grams.

      (a) Formulate the null and alternative hypotheses to test if a product weighing 540 grams is statistically significantly heavier than average.

      (b) Calculate the Z-score for a product weighing 540 grams.

      (c) Determine the probability (using the Z-table) of a product weighing above 540 grams.

      (d) If one million products are produced, estimate how many products are expected to exceed 540 grams.

   (b) (10 points) **Interpretation of Z-Scores:**

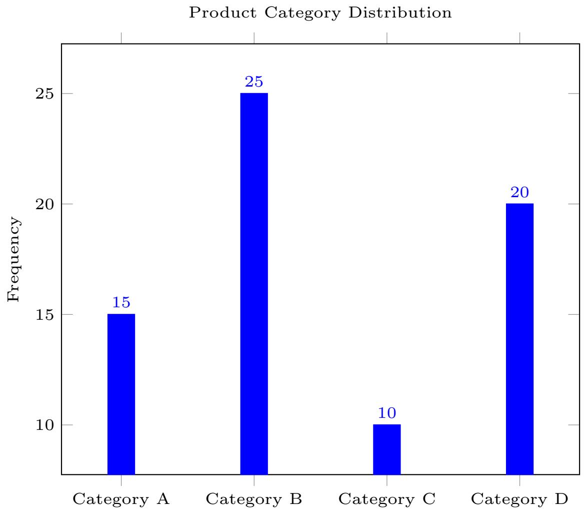
   - Provide a detailed explanation of how Z-scores standardize data.
   - Discuss two scenarios where comparing Z-scores (instead of raw values) is advantageous.

   (c) (10 points) **Standard Deviation Application in Decision Making:**

   - Explain how the standard deviation can affect business decisions when assessing product quality.
   - Provide an example scenario where a high standard deviation might indicate issues in production.

Data Visualization Analysis

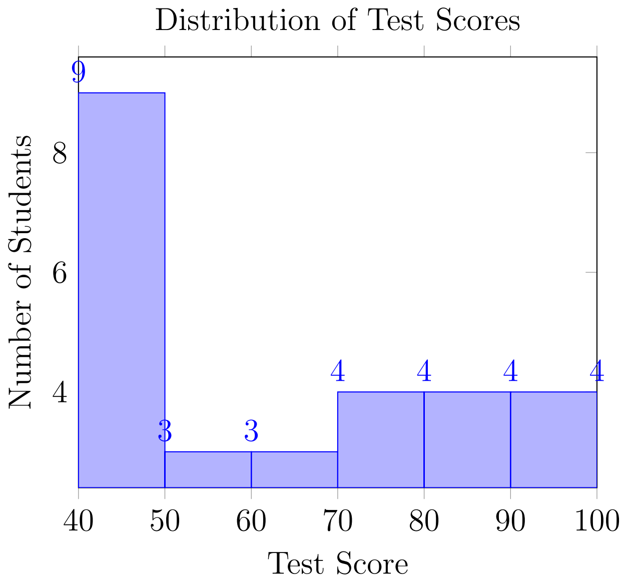
(a) Bar Graph: Product Categories



Analysis Questions:

1. **Data Type:** ________________________
2. **Measurement Level:** ________________________
3. **Describe the distribution of product categories:**
4. **Potential marketing insights from this distribution:**

(b) Histogram: Test Scores



Analysis Questions:

1. **Data Type:** ______________________
2. **Measurement Level:** ______________________
3. **Interpret the distribution of test scores:**

4. **Why is a histogram appropriate for this type of data?**

(c) Broader Understanding

1. **Explain how understanding data types and measurement levels guides the selection of proper visualization methods:**

3. Python Code Analysis and Output Prediction (25 points)

Below are 10 tricky Python code snippets. For each snippet, work out the expected output manually and record your results on paper. Approximately 10 output lines are expected overall. Provide a brief explanation for any non-obvious behavior.

For each of the following Python code snippets, carefully trace the code and determine:

- The complete output
- Explain any non-obvious behavior

4. Complex List Comprehension

```python
nums = [x * y for x in range(1, 6)
                for y in range(1, 6)
                if x * y % 3 == 0]
print(nums)
```

5. String Transformation

```python
s = "python programming"
result = ''.join([
    char.upper() if i % 2 == 0 else char.lower()
    for i, char in enumerate(s)
    if char != ' '
])
print(result)
```

6. Dynamic Dictionary Creation

```python
def is_prime(n):
    return n > 1 and all(n % i != 0 for i in range(2, int(n**0.5) + 1))

primes_dict = {x: is_prime(x) for x in range(2, 11)}
print(primes_dict)
```

7. Nested Loop with Break and Continue

```python
result = []
for i in range(1, 5):
    for j in range(i, 6):
        if i * j > 10:
            break
        if i + j < 4:
            continue
        result.append((i, j))
print(result)
```

8. Sequence Generation

```
def generate_sequence(n):
    return [
        sum(sequence[-2:])
        for sequence in [[1, 1] + [
            sum([sequence[-1], sequence[-2]])
            for _ in range(n-2)
        ] for sequence in [[1, 1]]]
    ][0]

print(generate_sequence(6))
```

9. Advanced String Slicing

```
s = "PYTHON"
results = (
    s[::2],  # Every second character
    s[::-1],  # Reversed string
    s[len(s)//2:] + s[:len(s)//2],  # Rotate middle
    ''.join(sorted(set(s.lower())))  # Unique sorted characters
)
print(results)
```

10. Multi-Condition List Filtering

```
filtered = [
    x for x in range(1, 30)
    if x % 2 == 0 and
       sum(int(digit) for digit in str(x)) % 3 == 0 and
       x > 10
]
print(filtered)
```

11. Tuple and Dictionary Comprehension

```
divisors_dict = {
    x: tuple(y for y in range(1, x+1) if x % y == 0)
    for x in range(1, 10)
    if x % 3 == 0
}
print(divisors_dict)
```

12. Conditional Expression Transformation

```
numbers = [-5, -2, 0, 3, 4, 7]
transformed = [
    x**2 if x % 2 == 0 else x+10
    if x > 0 else abs(x)
    for x in numbers
]
print(transformed)
```

13. **Nested Iteration with Filtering**

```python
complex_pairs = [
    (x, y)
    for x in range(1, 7)
    for y in range(x, 7)
    if x * y % 5 == 0 and x + y > 7
]
print(complex_pairs)
```

End of Exam.