复习题：

1、

（1）数据创建：数据的生成和采集阶段，包括数据输入、数据录入、数据采集等。

（2）数据存储：将数据存放在物理或虚拟存储设备中的过程。这一阶段涉及选择存储介质、数据库及其配置。

（3）数据使用：数据被访问、分析和处理的阶段，包括数据查询、报告生成、数据分析等。

（4）数据共享：将数据分发给其他用户或系统的过程，包括内部共享和外部共享。

（5）数据维护：对数据进行更新、清理和修正等维护操作，以确保数据的准确性和可用性。

（6）数据归档：将不再频繁使用的数据转移到长期存储的过程，以便释放主存储的空间，同时保留数据以备将来查阅。

（7）数据删除：在数据不再需要时进行安全删除的步骤，确保数据不被非法访问或恢复。

（8）数据审计和合规：确保数据管理符合相关法律法规和政策的过程，包括数据保护、隐私和合规性检查。

2、

数据采集是指从各种来源收集数据的过程，以便后续的分析、处理和利用。

方法：

1. 手动采集
2. 自动化采集
3. 二次数据采集
4. API数据采集
5. 数据流采集
6. 混合采集

3、

数据管理是指对数据的收集、存储、使用、共享和维护的系统化管理过程。其目的是确保数据的准确性、一致性、保密性以及可用性，以支持组织的决策和运营。

异同：

1. 数据类型和规模：

传统数据规模小且结构明确

大数据规模大且结构多样复杂

1. 数据储存：

传统数据管理一般使用关系型数据库（如Oracle、MySQL）来存储数据，强调数据的结构性和一致性。

大数据管理技术采用分布式存储系统（如Hadoop HDFS、NoSQL数据库如Cassandra、MongoDB等），能够处理海量数据并进行横向扩展。

1. 数据处理：

传统数据处理主要是通过ETL（提取、转换、加载）过程和SQL查询来完成，方式较为单一。

大数据管理技术使用多种处理框架（如Apache Hadoop、Apache Spark等），支持实时处理和批处理，能够进行复杂的数据分析和挖掘。

1. 数据更新和查询速度：

由于数据量较小，数据的更新和查询速度较快。

大数据管理技术由于数据量巨大及并发需求高，更新和查询的速度可能相对较慢，但采用分布式架构可以显著提高效率。

1. 数据分析：

传统数据管理分析模式较为简单，主要依靠预定义的报表和BI工具。

大数据管理技术采用先进的数据分析技术，包括机器学习、数据挖掘和实时分析，以挖掘数据中的潜在价值。

1. 数据安全

传统数据管理数据治理、合规性和安全策略相对成熟，容易获得明确的控制。

大数据管理技术由于数据来源多样，治理和安全面临更多挑战，需要制定复杂的法规和策略来维护数据的安全性和合规性。

4、

批处理，流处理，交互式查询、统一处理、图计算、机器学习。

5、

数据分析是对收集到的数据进行处理和解释的过程，以提取有价值的信息、识别模式、支持决策或预测未来趋势。

1. 描述性分析
2. 诊断性分析
3. 预测性分析
4. 规范性分析
5. 实时分析
6. 文本分析
7. 图分析

6、

1. 提高理解吸收能力
2. 识别模式和趋势
3. 支持决策制定
4. 强化沟通交流
5. 发现数据质量问题
6. 增加可访问性

践习题：

7、

import matplotlib.pyplot as plt

import numpy as np

days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

high\_temps = [30, 32, 28, 33, 31, 29, 34] # 最高气温

low\_temps = [20, 22, 19, 21, 23, 20, 24]# 最低气温

plt.figure(figsize=(10, 5))

plt.plot(days, high\_temps, label='High Temps', marker='o')

plt.plot(days, low\_temps, label='Low Temps', marker='x')

plt.title('Weekly Temperature Trends')

plt.xlabel('Day of the Week')

plt.ylabel('Temperature (°C)')

plt.xticks(days)

plt.legend()

plt.grid(True)

plt.show()

# 绘制柱状图

plt.figure(figsize=(10, 5))

bar\_width = 0.35

index = np.arange(len(days))

plt.bar(index, high\_temps, bar\_width, label='High Temps', color='skyblue')

plt.bar(index + bar\_width, low\_temps, bar\_width, label='Low Temps', color='lightcoral')

plt.title('Weekly High and Low Temperatures')

plt.xlabel('Day of the Week')

plt.ylabel('Temperature (°C)')

plt.xticks(index + bar\_width / 2, days)

plt.legend()

plt.grid(True)

plt.show()

# 绘制饼状图

temp\_ranges = [(20, 30), (30, 35)]

range\_counts = [sum(1 for temp in high\_temps if low\_bound <= temp < high\_bound) for low\_bound, high\_bound in temp\_ranges]

range\_labels = ['20-30°C', '30-35°C']

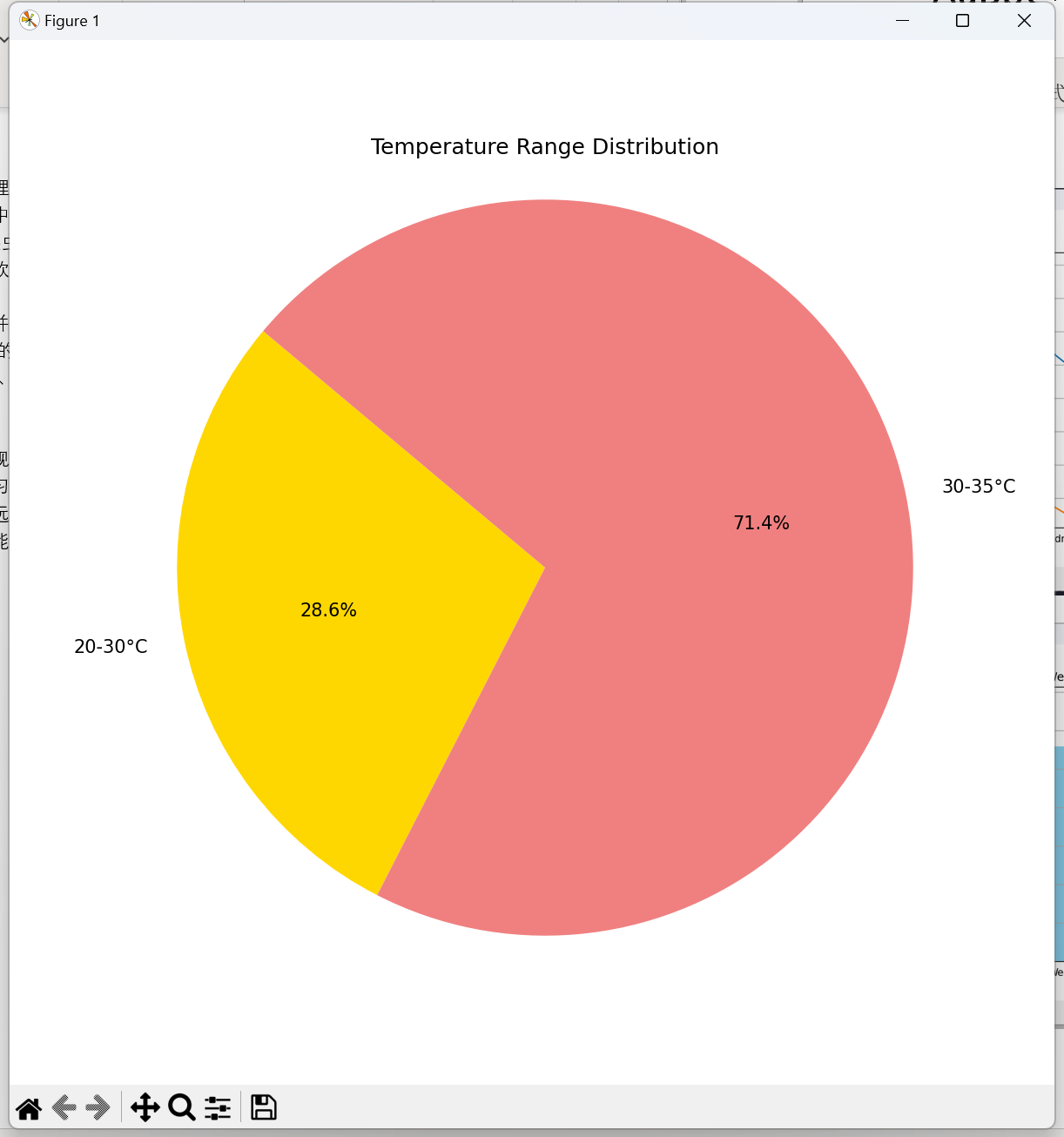
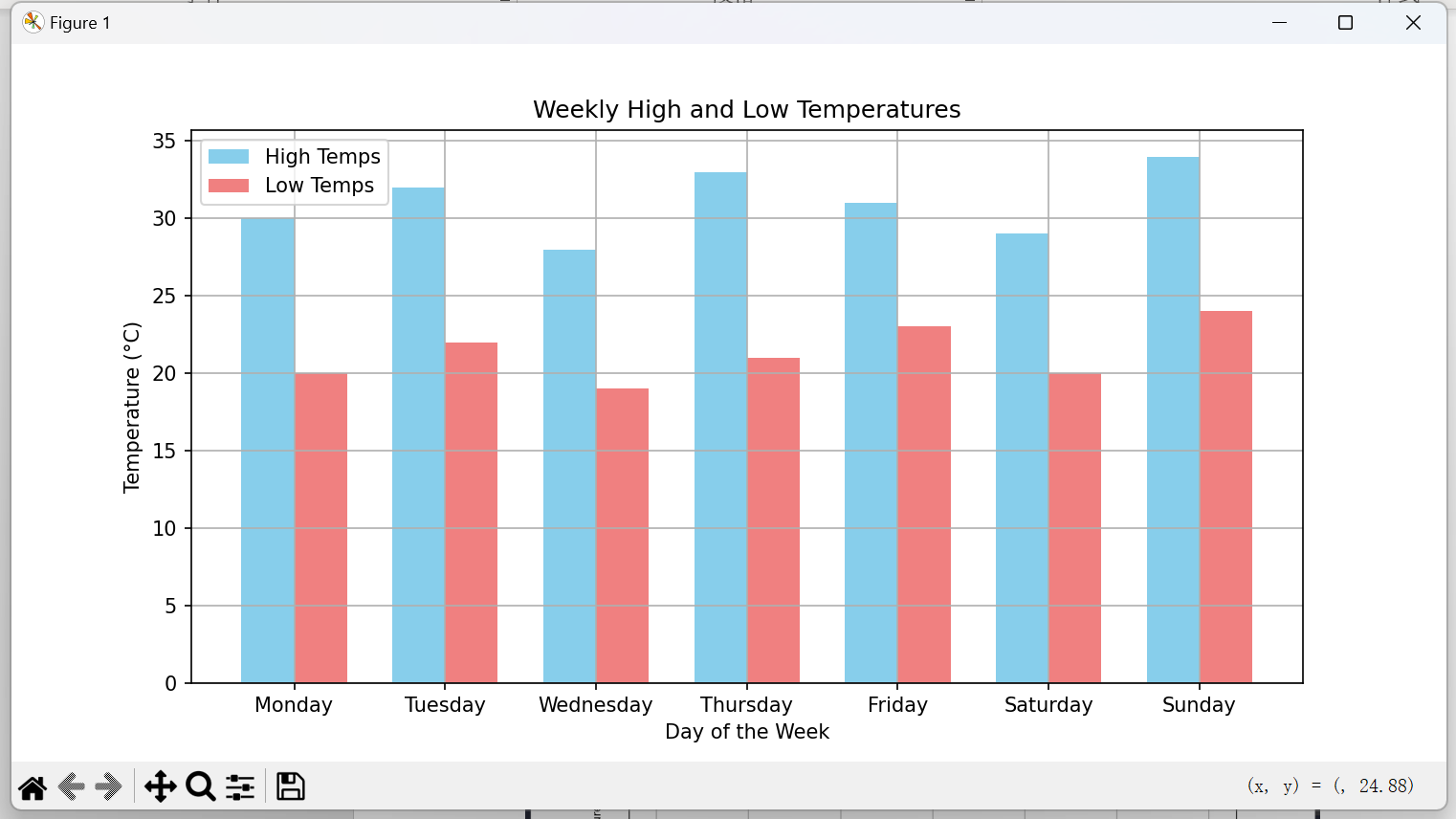
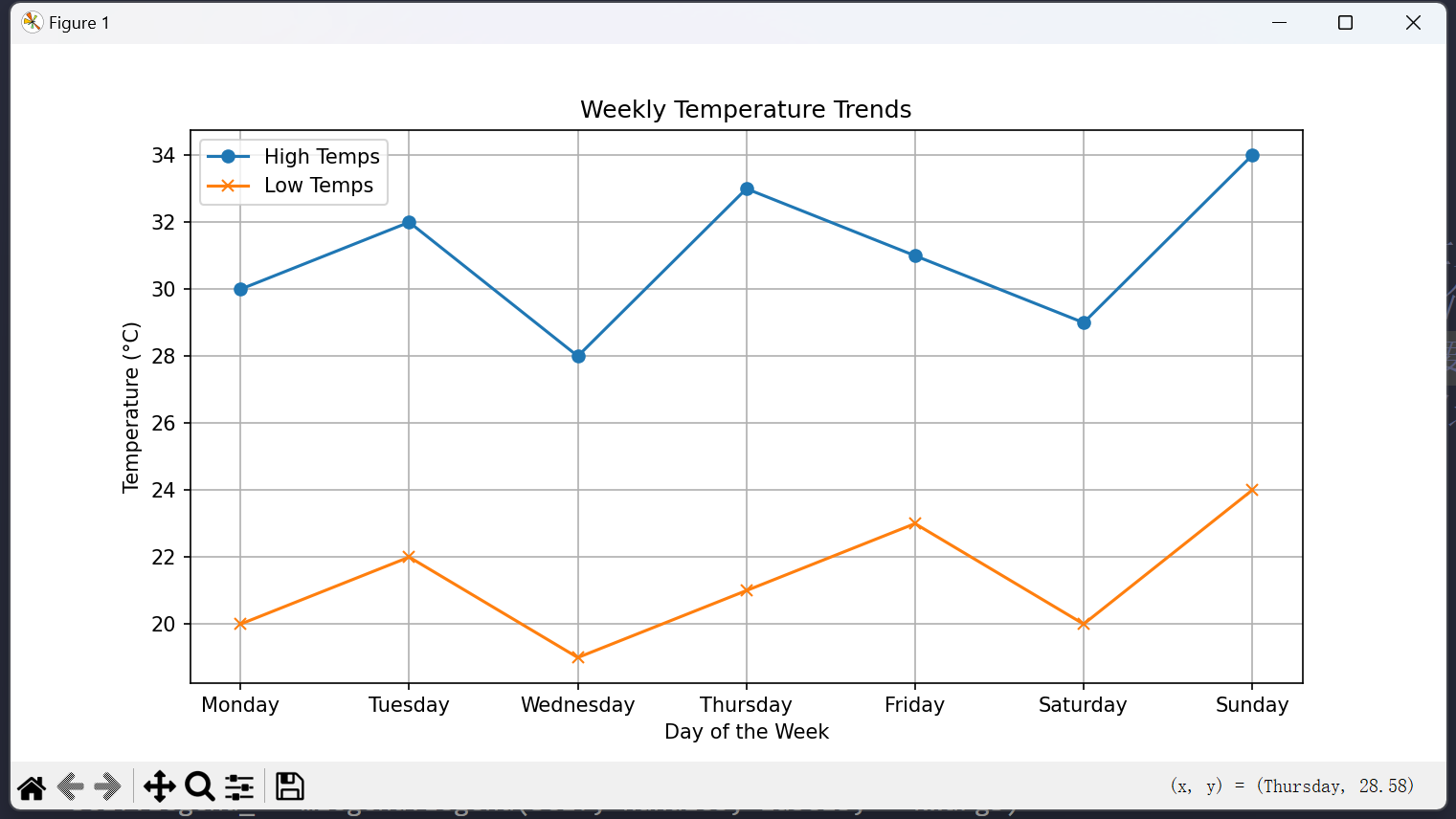
plt.figure(figsize=(8, 8))

plt.pie(range\_counts, labels=range\_labels, autopct='%1.1f%%', startangle=140, colors=['gold', 'lightcoral'])

plt.title('Temperature Range Distribution')

plt.axis('equal')

plt.show()



8、

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

import pandas as pd

# 假设的气温数据

days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

high\_temps = [30, 32, 28, 33, 31, 29, 34]  # 最高气温

low\_temps = [20, 22, 19, 21, 23, 20, 24]    # 最低气温

# 创建DataFrame

data = pd.DataFrame({

    'Day': days,

    'High Temps': high\_temps,

    'Low Temps': low\_temps

})

# 绘制折线图

plt.figure(figsize=(10, 5))

sns.lineplot(x='Day', y='High Temps', data=data, marker='o', label='High Temps', color='blue')

sns.lineplot(x='Day', y='Low Temps', data=data, marker='x', label='Low Temps', color='red')

plt.title('Weekly Temperature Trends')

plt.xlabel('Day of the Week')

plt.ylabel('Temperature (°C)')

plt.xticks(rotation=45)

plt.legend()

plt.grid(True)

plt.show()

# 绘制柱状图

plt.figure(figsize=(10, 5))

bar\_width = 0.35

index = np.arange(len(days))

sns.barplot(x='Day', y='High Temps', data=data, color='skyblue', label='High Temps', alpha=0.7)

sns.barplot(x='Day', y='Low Temps', data=data, color='lightcoral', label='Low Temps', alpha=0.7)

plt.title('Weekly High and Low Temperatures')

plt.xlabel('Day of the Week')

plt.ylabel('Temperature (°C)')

plt.legend()

plt.grid(True)

plt.show()

# 绘制饼状图

temp\_ranges = [(20, 30), (30, 35)]

range\_counts = [sum(1 for temp in high\_temps if low\_bound <= temp < high\_bound) for low\_bound, high\_bound in temp\_ranges]

range\_labels = ['20-30°C', '30-35°C']

plt.figure(figsize=(8, 8))

plt.pie(range\_counts, labels=range\_labels, autopct='%1.1f%%', startangle=140, colors=['gold', 'lightcoral'])

plt.title('Temperature Range Distribution')

plt.axis('equal')  # 使饼状图为圆形

plt.show()

