1.

a). Rdd=spark.read.option(header,”true”).option(inferSchema,”True”).csv(“table_path”)

b). df1.select("case_id","province","infection_case")

c). df1.show()

2.

```
from pyspark.sql import SparkSession

# Create a SparkSession
spark = SparkSession.builder.appName("DataFrameExample").getOrCreate()

# Load CSV file into a DataFrame
df = spark.read.csv("file:///path/to/your/file.csv", header=True, inferSchema=True)

# Show the first few rows of the DataFrame
df.show()

# Perform filtering on the DataFrame
filtered_df = df.filter(df["column_name"] > 10)

# Group by a column and compute aggregate functions
grouped_df = df.groupBy("column_name").agg({"other_column": "sum", "another_column": "avg"})

# Join two DataFrames
df1 = df.select("column_name1", "column_name2")
df2 = df.select("column_name3", "column_name4")
joined_df = df1.join(df2, df1["column_name1"] == df2["column_name3"], "inner")

# Apply Spark SQL queries on the DataFrame
df.createOrReplaceTempView("my_table")
```

```
result = spark.sql("SELECT column_name1, AVG(column_name2) FROM my_table WHERE column_name3 > 100 GROUP BY column_name1")

# Show the result of the SQL query
result.show()

# Stop the SparkSession
spark.stop()
```

3.

```
from pyspark.sql import SparkSession

# Connect Spark with a relational database (MySQL or PostgreSQL)
def connect_to_database(database_type, host, port, database, username, password):
    spark = SparkSession.builder \
        .appName("Spark-Database Integration") \
        .config("spark.driver.extraClassPath", "path_to_jdbc_driver.jar") \
        .getOrCreate()

    url = f"jdbc:{database_type}://{host}:{port}/{database}"
    properties = {
        "user": username,
        "password": password
    }

    return spark.read.jdbc(url=url, table="table_name", properties=properties)

# Perform SQL operations on the data stored in the database using Spark SQL
def perform_sql_operations(dataframe):
    dataframe.createOrReplaceTempView("my_table")
```

```
    # Example SQL query
    result = spark.sql("SELECT * FROM my_table WHERE column_name = 'value'")

    # Display the result
    result.show()

# Explore integration capabilities with other data sources (HDFS or Amazon S3)
def read_data_from_hdfs(file_path):
    # Read data from HDFS
    hdfs_df = spark.read.text(file_path)
    hdfs_df.show()

def read_data_from_s3(bucket_name, file_key):
    # Read data from Amazon S3
    s3_df = spark.read.text(f"s3a://{bucket_name}/{file_key}")
    s3_df.show()

# Configuration for connecting to the database
database_type = "mysql"  # or "postgresql"
host = "localhost"
port = "3306"  # MySQL default port: 3306, PostgreSQL default port: 5432
database = "your_database_name"
username = "your_username"
password = "your_password"

# Connect to the database
dataframe = connect_to_database(database_type, host, port, database, username, password)

# Perform SQL operations
perform_sql_operations(dataframe)
```

```
# Read data from HDFS
hdfs_file_path = "hdfs://localhost:9000/path_to_file.txt"
read_data_from_hdfs(hdfs_file_path)

# Read data from Amazon S3
s3_bucket_name = "your_bucket_name"
s3_file_key = "path_to_file.txt"
read_data_from_s3(s3_bucket_name, s3_file_key)
```