Let's break down each topic and provide a sample PySpark code for each.

19. Broadcast Variables and Accumulators:

Broadcast Variables:
Broadcast variables allow the efficient sharing of read-only variables across all nodes in a distributed Spark environment. These variables are cached on each machine rather than being sent over the network with tasks.

```python
from pyspark import SparkContext, SparkConf

conf = SparkConf().setAppName("BroadcastExample")
sc = SparkContext(conf=conf)

# Create a broadcast variable
broadcast_var = sc.broadcast([1, 2, 3, 4, 5])

# Use the broadcast variable in a Spark transformation
rdd_data = sc.parallelize([10, 20, 30, 40, 50])
result = rdd_data.map(lambda x: x * broadcast_var.value[0])

# Collect and print the result
print(result.collect())

sc.stop()
```

Accumulators:
Accumulators are variables that can be added to by associative and commutative operations and are used to implement counters and sums in a distributed manner.

```python
from pyspark import SparkContext, SparkConf

conf = SparkConf().setAppName("AccumulatorExample")
sc = SparkContext(conf=conf)

# Create an accumulator variable
accumulator_var = sc.accumulator(0)

# Use the accumulator variable in a Spark transformation
rdd_data = sc.parallelize([1, 2, 3, 4, 5])
rdd_data.foreach(lambda x: accumulator_var.add(x))
```

```
# Print the value of the accumulator
print(accumulator_var.value)

sc.stop()
```

20. Handling Missing Data:

Imputation Strategies:
Imputation is the process of replacing missing values with substituted values. PySpark provides various imputation functions.

```python
from pyspark.sql import SparkSession
from pyspark.ml.feature import Imputer

spark = SparkSession.builder.appName("MissingDataExample").getOrCreate()

# Create a DataFrame with missing values
data = [(1, None), (2, 5), (3, 10), (4, None)]
columns = ["id", "value"]
df = spark.createDataFrame(data, columns)

# Use Imputer to fill missing values with mean
imputer = Imputer(inputCols=["value"], outputCols=["value_imputed"])
model = imputer.fit(df)
df_imputed = model.transform(df)

# Show the DataFrame with imputed values
df_imputed.show()

spark.stop()
```

Dropping and Filling Missing Values:
You can also drop or fill missing values using DataFrame operations.

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

spark = SparkSession.builder.appName("MissingDataExample").getOrCreate()
```

```
# Create a DataFrame with missing values
data = [(1, None), (2, 5), (3, 10), (4, None)]
columns = ["id", "value"]
df = spark.createDataFrame(data, columns)

# Drop rows with missing values
df_dropped = df.na.drop()

# Fill missing values with a specific value
df_filled = df.na.fill(0, subset=["value"])

# Show the DataFrames
df_dropped.show()
df_filled.show()

spark.stop()
```

These examples should help you understand how to use broadcast variables, accumulators, and handle missing data in PySpark. Make sure to adapt them to your specific use case and data.