Assume you have full knowledge on SDET and having best experience in answering in interview then "what is equivalence partitioning, boundary value analysis, decision table testing, state transition testing? with examples"

Absolutely, Routh Kiran Babu. Below is a professional, **interview-level answer** for each testing technique you're asking about — **Equivalence Partitioning**, **Boundary Value Analysis**, **Decision Table Testing**, and **State Transition Testing** — along with **clear examples**. This will help you **answer confidently as an SDET** in any interview.

✅ 1. Equivalence Partitioning (EP)

🔹 Definition:

Equivalence Partitioning is a **black-box testing technique** where input data is divided into **partitions (classes)** of **valid and invalid inputs**, and only **one value from each partition is tested**. It helps reduce the number of test cases while maintaining coverage.

✅ Why use it?

To avoid testing all values and still ensure good test coverage.

🔸 Example:

Assume a field accepts age between **18 to 60**.

Partition Type	Partition Description	Example Value
Valid	18 ≤ Age ≤ 60	25
Invalid	Age < 18	15
Invalid	Age > 60	65

➡ Instead of testing every number, we pick **one from each class**: 15, 25, and 65.

✅ 2. Boundary Value Analysis (BVA)

🔹 Definition:

BVA is based on testing **edge/boundary values** because errors often occur at boundaries of input ranges rather than the center.

✅ Why use it?

Most defects occur near boundary conditions.

🔸 Example:

Same field: age must be between **18 and 60** (inclusive).

Condition	Test Values
Just below min	17
Min value	18
Just above min	19
Just below max	59
Max value	60
Just above max	61

➡ **Test Cases**: `17`, `18`, `19`, `59`, `60`, `61`

✅ 3. Decision Table Testing

🔹 Definition:

A decision table is used for testing systems with **complex business logic** involving **combinations of inputs** to produce **different outputs (actions/decisions)**.

✅ Why use it?

To ensure **all possible input combinations** are tested especially when dealing with rules/conditions.

🔸 Example:

Suppose a login form requires:

- **Condition 1**: Valid username?
- **Condition 2**: Valid password?
- **Action**: Grant access or Deny

TC	Condition 1 (Username)	Condition 2 (Password)	Action
1	T	T	Grant Access
2	T	F	Deny Access
3	F	T	Deny Access
4	F	F	Deny Access

➡ Decision Table helps us visualize combinations and ensure nothing is missed.

✅ 4. State Transition Testing

🔹 Definition:

This technique is used when the system **changes its behavior depending on current state and input**. It models the software as a **finite state machine**.

✅ Why use it?

To test scenarios where **past actions (states)** affect current behavior.

🔸 Example: ATM Machine

Current State	Input Action	Next State	Output
Card Inserted	Enter PIN	Authenticated	Ask for Transaction
Authenticated	Withdraw Money	Dispensing	Dispense Cash
Authenticated	Cancel	Eject Card	Return Card
Card Inserted	Wrong PIN (3x)	Blocked	Block Card

➡ Test transitions from one state to another with the correct input and expected outcome.

✅ Summary Table

Technique	Purpose	Best Used When
Equivalence Partitioning	Reduce test cases by grouping inputs	Input fields with valid/invalid ranges
Boundary Value Analysis	Test edge values	Numerical inputs with boundaries
Decision Table Testing	Handle complex business rules	Multiple input combinations
State Transition Testing	Test state-based systems	Systems with sequential states

---

If you're in an **SDET interview**, you can say:

> “I use Equivalence Partitioning and BVA during test case design to ensure coverage with minimal effort. For example, when validating age input fields or form ranges. When the logic is rule-based, I switch to Decision Table Testing to cover all combinations. And for applications like login workflows or ATMs, I prefer State Transition Testing to ensure correct behavior across various state flows.”

Let me know if you'd like a **Java or Python code example** for any of the above or how to **automate these test scenarios** using Cypress, Selenium, or Rest Assured.