A] React.createElement :

1. React createElement API returns a React element object with a few properties.
   a. **Type** - type we have passed
   b. **props** - The props you have passed except for ref and key.
   c. **ref** - The ref you have passed. If missing, null.
   d. **key** - The key you have passed, coerced to a string. If missing, null.

2. Parameters
   1. **type** **(string | React.createClass()):-**
      - must be a valid React component type (html element tag wrapped in stringor React component)
   2. **Props** **(null | object):-**
      - object or null. null equivalents to empty object. we can add attributes to our tag in this object.
   3. **Children** **(null | string | React.createClass() | React.createElement()): -**
      - Zero or moree child nodes.they can be React Nodes, including React elements, strings, numbers, portals, empty nodes (null, undefined, true, and false), and arrays of React nodes.
   4. Following are the standard HTML elements that React supports (i.e. these elements passed as a string `type` to `createElement()` will create the associating standard HTML element in the DOM):
      `a abbr address area article aside audio b base bdi bdo big blockquote body brbutton canvas caption cite code col colgroup data datalist dd del details dfn dialog div dl dt em embed fieldset figcaption figure footer form h1 h2 h3 h4 h5 h6 head header hgroup hr html i iframe img input ins kbd keygen label legend li link main map mark menu menuitem meta meter nav noscript object ol optgroup option output p param picture pre progress q rp rt ruby s samp script section select small source span strong style sub summary sup table tbody td textarea tfoot th thead time title tr track u ul var video wbr`

3. React Element :
   - an element is a lightweight description of a piece of the user interface.
   - Creating elements is extremely cheap so you don't need to try to optimize or avoid it.
   - the above headerElement return a React element which when logged looks like
   - eg.

```
{
   type: 'h1',
   props: {
      children: 'Hello World from React',
      id: "main"
   },
   key: null,
```

ref: null,
}

B] root.render(reactNode):

1. Call root.render to display a piece of JSX (“React node”) into the React root’s browser DOM node.
2. Parameters :
   - **reactNode :**
     - A React node that you want to display. This will usually be a piece of JSX like <App />, but you can also pass a React element constructed with **createElement()**, a string, a number, null, or undefined.
3. Returns :
   - undefined
4. If you call render on the same root more than once, React will update the DOM as necessary to reflect the latest React Node you passed.

C] What is a React Node?

1. The primary type or value that is created when using react is known as React Node. A React node is defined as
   A light, stateless, immutable, virtual representation of a DOM node.
2. React nodes are not real DOM node themselves but a representation of a potential DOM node. That repersentation is considered the **virtual DOM**.
3. React is used to define a virtual DOM using React Nodes, that fuel the React Components, that can eventually be used to create real DOM structure or other structures (e.g., React Native).
4. React Nodes can be created using **JSX** or **JavaScript(createElement).**

D] What is Virtual DOM?

1. A virtual DOM is a lightweight JavaScript object which is the copy of the real DOM.
2. This Virtual DOM works in three simple steps-
   - Whenever any underlying data changes, the entire UI is re-rendered in Virtual DOM representation
   - Then the difference between the previous DOM representation and the new one is calculated
   - Once the calculations are done, the real DOM will be updated with only the things that have actually changed

E] ReactDOM.render():

1. render() function can be used to render React Nodes to the Real DOM.
2. Render function converts React Nodes to Virtual DOM ane then to real HTML DOM.
3. Any DOM nodes inside of the HTML DOM element (<div id=”root”></div>) which you are rendering into will be replaced (i.e., removed).
4. Re-rendering to the same DOM element will only update the current child nodes if a change has been made or a new child node has been added.

References :

1. https://react.dev/reference/react/createElement
2. https://react.dev/reference/react-dom/client/createRoot
3. https://www.reactenlightenment.com/react-nodes/4.1.html
4. Google_Doc_Link :
   https://docs.google.com/document/d/1nk_NH2SjvyBg6I1Je_YVSnaOhLOD9WW3rwjE9MhBcaM/edit?usp=sharing