1. Including Libraries

```
#include <Trade/Trade.mqh>
CTrade Trader;
```

Purpose:

- The `Trade.mqh` library is included to simplify trading operations like opening, closing, and managing positions.
- `CTrade` is an object that encapsulates these trading functionalities, and `Trader` is the instance used throughout the code.

2. Indicator and Input Parameters

```
int RSIhandler;
input int ma_period;
input ENUM_APPLIED_PRICE applied_price;
input double lot_size;
int barsTotal;
int bars;
bool trade = true;
input int Position_Limiter;
```

Purpose:

- RSIhandler: Stores the handle for the RSI indicator.
- Inputs:
  - `ma_period`: The period used for the RSI calculation.
  - `applied_price`: Specifies the price type (e.g., close price, open price) used for the RSI.
  - `lot_size`: Defines the trade size (volume).
  - `Position_Limiter`: Limits the number of open trades at a time.
- Other variables:
  - `barsTotal`: Tracks the total number of bars to detect new ones.
  - `bars`: Used to compare with `barsTotal` to ensure trading logic runs only on new bars.
  - `trade`: A flag that controls whether new trades can be opened.

3. Trade Opener

```
//  A FUNCTION THAT OPENS THE TRADE
void Trade_Opener(){

   double RSILine[];
   CopyBuffer(RSIhandler,MAIN_LINE,0,1,RSILine);

   if(barsTotal == bars && trade){
      if(RSILine[0] < 20){
         Trader.Buy(lot_size);
      }
      else if(RSILine[0] > 80){
         Trader.Sell(lot_size);
      }
   }
}
```

Purpose:

- Opens trades based on RSI levels:
  - **Buy**: When RSI < 20 (oversold).
  - **Sell**: When RSI > 80 (overbought).
- Ensures trades are opened only on new bars (`barsTotal == bars`) and if trading is allowed (`trade` is `true`).

4. Trade Closer

```
// A FUNCTION THAT CLOSES THE TRADE
void Trade_Closer(){

   double RSILine[];
   CopyBuffer(RSIhandler,MAIN_LINE,0,1,RSILine);

   if(RSILine[0] <= 50){
      for(int i = PositionsTotal() - 1; i >= 0; i--){
         ulong positionTicket = PositionGetTicket(i);
         if(PositionSelectByTicket(positionTicket)){
            if(PositionGetInteger(POSITION_TYPE) == POSITION_TYPE_SELL){
               Trader.PositionClose(positionTicket);
            }
         }
      }
   }
```

```
   else if(RSILine[0] >= 50){
      for(int i = PositionsTotal() - 1; i >= 0; i--){
         ulong positionTicket = PositionGetTicket(i);
         if(PositionSelectByTicket(positionTicket)){
            if(PositionGetInteger(POSITION_TYPE) == POSITION_TYPE_BUY){
               Trader.PositionClose(positionTicket);
            }
         }
      }
   }
}
```

Purpose:

- Closes trades based on RSI values:
  - Close Sell Positions: When RSI ≤ 50.
  - Close Buy Positions: When RSI ≥ 50.
- Iterates through all open positions to identify and close those matching the criteria.

5. Trade Limiter

```
// A LIMIT TO THE NUMBER OF TRADES OPENED/OPENING
void Trade_Limiter(){

   int TradeLimit = PositionsTotal();

   if(TradeLimit == Position_Limiter){
      trade = false;
   }
   else{
      trade = true;
   }
}
```

Purpose:

- Restricts the number of open trades to the value specified in `Position_Limiter`.
- If the limit is reached, the `trade` flag is set to `false`, preventing new trades from opening.

6. OnInit

```
// THE ONINIT FUNCTION
int OnInit(){

   RSIhandler = iRSI(Symbol(),PERIOD_CURRENT,ma_period,applied_price);

   return(INIT_SUCCEEDED);
}
```

Purpose:

- Initializes the RSI indicator using the provided period and price type.
- Stores the indicator handle (`RSIhandler`) for use in trading functions.

7. OnTick

```
// THE ONTICK FUNCTION
void OnTick(){

   bars = iBars(Symbol(), PERIOD_CURRENT);
   if(barsTotal == bars) return;
   barsTotal = bars;

// THE TRADE FUNCTIONS ARE CALLED HERE
   Trade_Opener();
   Trade_Limiter();
   Trade_Closer();
}
```

Purpose:

- The main execution loop:
  - Detects new bars by comparing `bars` and `barsTotal`.
  - Calls the trade functions (`Trade_Opener`, `Trade_Limiter`, `Trade_Closer`) to manage trading activity.