

# Stock Market Predictions

---



## Phase 1: Problem Understanding & Industry Analysis

### 1. Requirement Gathering

The goal is to help investors and analysts manage stock data inside Salesforce.

- Store stock details (company name, ticker, sector, price).
- Allow investors to track their portfolios (buy/sell transactions).
- Show predictions (future price trends).
- Provide reports and dashboards (gains/losses, prediction accuracy).
- Send alerts when stock crosses certain thresholds.

### 2. Stakeholder Analysis

- Investors (End Users) → Want to see their portfolio and stock predictions.
- Financial Analysts → Need dashboards to compare stock performance.
- Admin/Developer → Setup objects, flows, automation, and integrations.
- Executives → Want summary dashboards for decision making.

### 3. Business Process Mapping

- Current Process (outside Salesforce):
  - Investors use multiple apps like Yahoo Finance, Excel, TradingView.
  - Predictions are either manual or from third-party websites.
  - No single system to combine portfolio + real-time data + predictions.
- Proposed Process (inside Salesforce):
  - Salesforce fetches stock data from an external API.
  - Data is stored in custom objects (Stock, Portfolio, Transaction, Prediction).

- Automation updates portfolio value and prediction records.
- Reports and dashboards show insights (Top gainers, Loss %).
- Alerts sent to investors when price moves significantly.

## 4. Industry-Specific Use Case Analysis

- Retail Investors → Manage their personal stock portfolios inside Salesforce.
- Investment Firms → Track multiple client portfolios in one platform.
- Wealth Management Companies → Give AI-based recommendations using predictions.
-  Example: If an investor has 10 shares of Company, Salesforce shows:
- Current Value = Quantity × Price.
- Predicted Next Price = +5%.
- System sends an alert: “Your Company holdings may rise 5% tomorrow.”

## 5. AppExchange Exploration

- Existing apps provide stock tickers or financial dashboards.
- But there is no complete prediction-focused app on AppExchange.
- This project fills that gap by combining portfolio tracking + prediction + alerts.



## Phase 2: Org Setup & Configuration

### 1. Salesforce Edition

- Use Developer Edition → It's free and provides all the features required (custom objects, automation, API integration).
- Suitable for learning and building a POC (Proof of Concept).

### 2. Company Profile Setup

- Fiscal Year → Set from January to December (to align with global stock reporting).
- Default Currency → Indian Rupee (₹) or USD (\$), depending on stock market focus.
- Timezone → Align with the stock exchange (e.g., GMT+5:30 for India).

### 3. Business Hours & Holidays

- Set Business Hours → 9:30 AM – 3:30 PM (Indian NSE/BSE market).
- Configure Holidays → Official stock market holidays (e.g., Diwali, Independence Day, Republic Day).
- Purpose → Helps with SLA calculations and alerts only during market hours.

### 4. User Setup & Licenses

- Users:
    - Admin User → Full control (manages configurations).
    - Investor User → Restricted access (views only their portfolio).
- Assign Salesforce Platform License to Investor user (cost-effective in real scenario).

### 5. Profiles

- Admin Profile → CRUD (Create, Read, Update, Delete) on all objects.
- Investor Profile → Read-only access to Stocks & Predictions, Read/Write to their Portfolio.

## **6. Roles**

- Admin Role → Higher in hierarchy, can see all data.
- Investor Role → Lower in hierarchy, can only see their own records.

## **7. Permission Sets**

- Create "API Access Permission Set" → Grants API access for stock data integration.
- Assign this only to Admin.

## **8. Org-Wide Defaults (OWD)**

- Portfolio & Transactions → Private (only owner can see).
- Stock → Public Read-Only (everyone can see stock prices).
- Prediction → Controlled by Parent (linked to Stock).

## **9. Sharing Rules**

- Share Portfolios only with the specific investor.
- Admin can see everything, but one investor cannot see another's portfolio.

## **10. Login Access Policies**

- Enable IP Restrictions for Admin login (e.g., office network only).
- Allow investors to log in from anywhere.

## **11. Dev Org Setup**

- Create Developer Org (from Salesforce.com → Free Signup).
- Install sample financial datasets (CSV imports).

## **12. Sandbox Usage**

Use Developer Sandbox for:

- Testing API integrations (stock price API).
- Testing automation (flows, triggers).

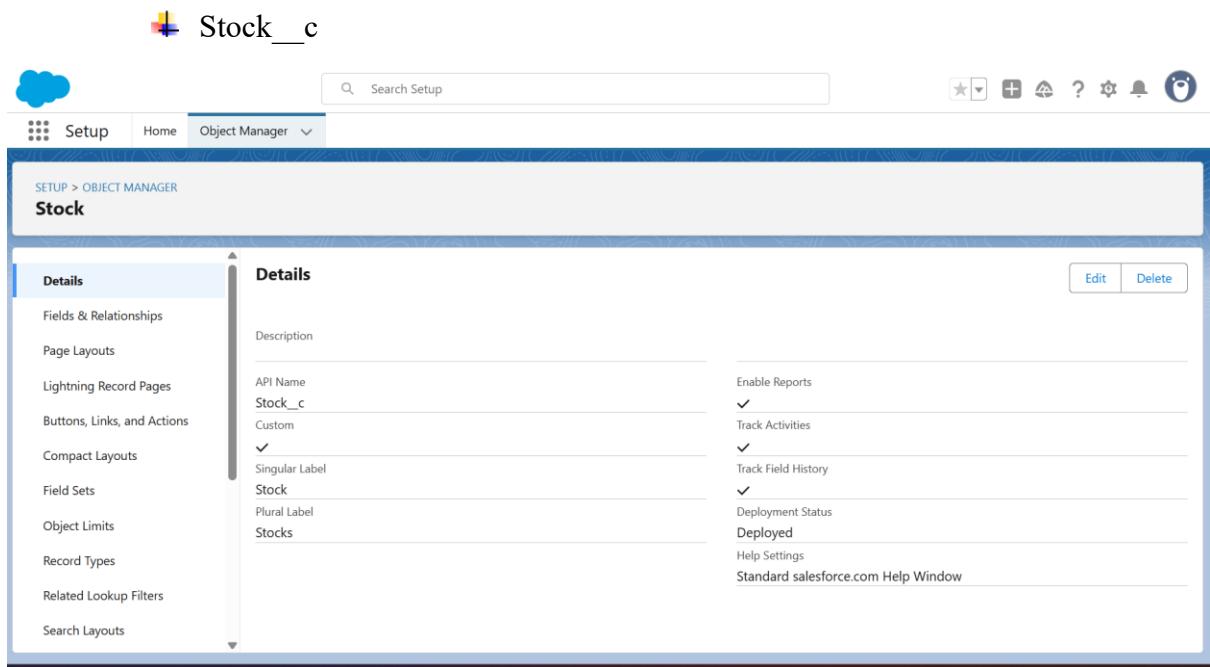
- Deploy to production only after validation.

## 13. Deployment Basics

- Use **Change Sets** for deployment (simple beginner method).
- Advanced option: Use **VS Code + SFDX** for metadata deployment.

# Phase 3: Data Modeling & Relationships

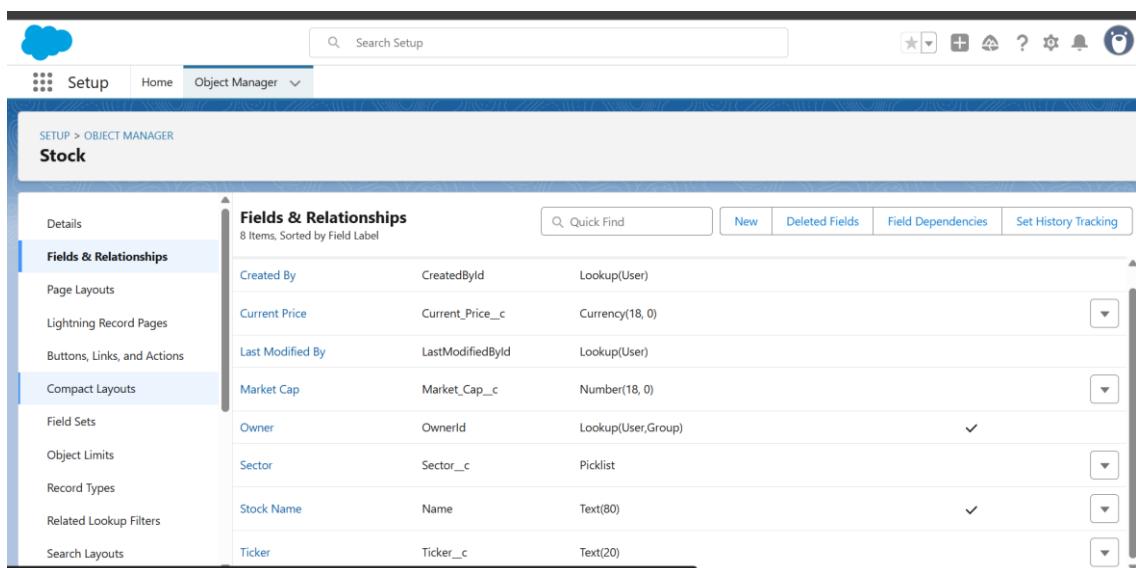
## 1. Custom Objects & Key Fields



The screenshot shows the Salesforce Object Manager interface for the Stock\_\_c object. The left sidebar lists various configuration options: Details, Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, and Search Layouts. The main pane displays the object's details, including its API name (Stock\_\_c), custom status (Custom), singular label (Stock), plural label (Stocks), and deployment status (Deployed). It also shows reporting and tracking settings like Enable Reports, Track Activities, and Track Field History.

### Fields:

- Ticker\_\_c (Text, e.g., TCS, INFY)
- Sector\_\_c (Picklist: IT, Pharma, Banking, etc.)
- Current\_Price\_\_c (Currency)
- Market\_Cap\_\_c (Number)



The screenshot shows the Salesforce Object Manager interface for the Stock\_\_c object, specifically the Fields & Relationships section. The left sidebar shows the same list of configuration options as the previous screenshot. The main pane lists the fields and their properties:

- Created By: CreatedById, Lookup(User)
- Current Price: Current\_Price\_\_c, Currency(18, 0)
- Last Modified By: LastModifiedById, Lookup(User)
- Market Cap: Market\_Cap\_\_c, Number(18, 0)
- Owner: OwnerId, Lookup(User,Group)
- Sector: Sector\_\_c, Picklist
- Stock Name: Name, Text(80)
- Ticker: Ticker\_\_c, Text(20)

## Portfolio\_c

Setup > OBJECT MANAGER  
**Portfolio**

**Details**

Description

API Name  
**Portfolio\_c**

Custom  
✓

Singular Label  
**Portfolio**

Plural Label  
**Portfolios**

Enable Reports  
✓

Track Activities  
✓

Track Field History  
✓

Deployment Status  
**Deployed**

Help Settings  
Standard salesforce.com Help Window

Edit Delete

### Fields:

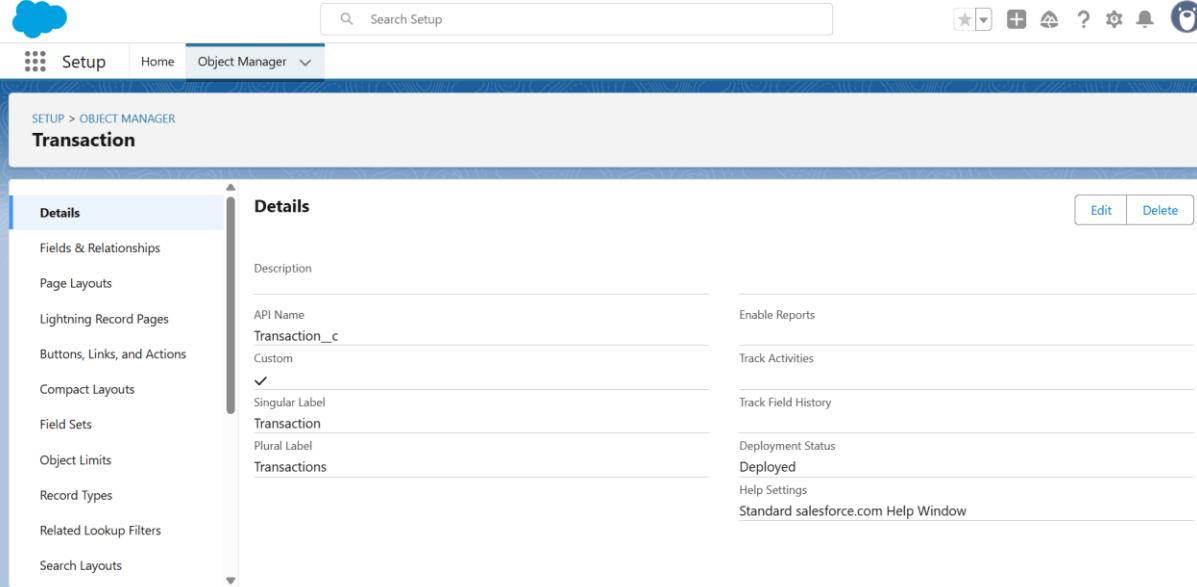
- Investor\_\_c (Lookup → User)
- Total\_Value\_\_c (Currency, roll-up from Transactions)
- Risk\_Profile\_\_c (Picklist: Low, Medium, High)

Setup > OBJECT MANAGER  
**Portfolio**

**Fields & Relationships**  
7 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Investor	Investor__c	Lookup(User)		✓
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Portfolio Name	Name	Text(80)		✓
Risk Profile	Risk_Profile__c	Picklist		
Total Value	Total_Value__c	Currency(18, 0)		

## Transaction\_\_c (Junction between Stock & Portfolio)



SETUP > OBJECT MANAGER  
**Transaction**

**Details**

Description

API Name  
**Transaction\_\_c**

Custom  
✓

Singular Label  
**Transaction**

Plural Label  
**Transactions**

Enable Reports

Track Activities

Track Field History

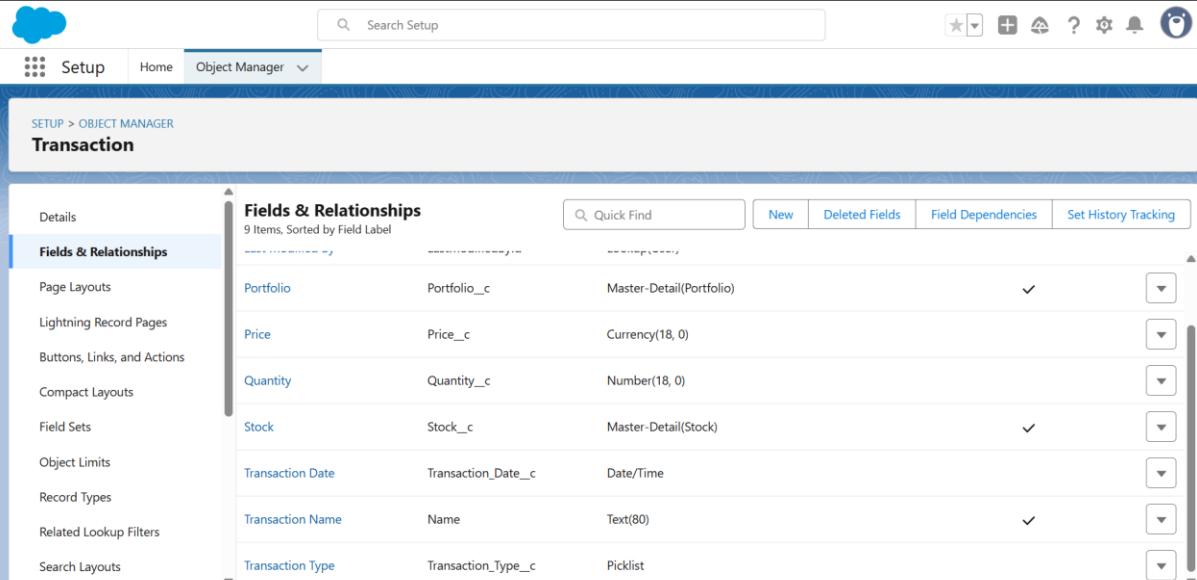
Deployment Status  
Deployed

Help Settings

Standard salesforce.com Help Window

### Fields:

- Portfolio\_\_c (Master-Detail)
- Stock\_\_c (Master-Detail)
- Transaction\_Type\_\_c (Picklist: Buy, Sell)
- Quantity\_\_c (Number)
- Price\_\_c (Currency)
- Transaction\_Date\_\_c (Date/Time)



SETUP > OBJECT MANAGER  
**Transaction**

**Fields & Relationships**  
9 Items. Sorted by Field Label

Portfolio	Portfolio__c	Master-Detail(Portfolio)	▼
Price	Price__c	Currency(18, 0)	▼
Quantity	Quantity__c	Number(18, 0)	▼
Stock	Stock__c	Master-Detail(Stock)	▼
Transaction Date	Transaction_Date__c	Date/Time	▼
Transaction Name	Name	Text(80)	▼
Transaction Type	Transaction_Type__c	Picklist	▼

## Prediction\_\_c

**Object Manager**

**Prediction**

**Details**

Description

API Name  
**Prediction\_\_c**

Custom ✓

Singular Label  
Prediction

Plural Label  
Predictions

Enable Reports ✓

Track Activities ✓

Track Field History ✓

Deployment Status  
Deployed

Help Settings

Standard salesforce.com Help Window

**Fields & Relationships**

- Page Layouts
- Lightning Record Pages
- Buttons, Links, and Actions
- Compact Layouts
- Field Sets
- Object Limits
- Record Types
- Related Lookup Filters
- Search Layouts

- Fields:
- Stock\_\_c (Lookup)
- Predicted\_Price\_\_c (Currency)
- Confidence\_Score\_\_c (Percent)
- Prediction\_Date\_\_c (Date)
- Prediction\_Model\_\_c (Picklist: Moving Avg, ML, AI Service, etc.)

**Fields & Relationships**

9 Items, Sorted by Field Label

Field	Type	Description
Created By	CreatedById	Lookup(User)
Last Modified By	LastModifiedById	Lookup(User)
Owner	OwnerId	Lookup(User,Group)
Predicted Price	Predicted_Price__c	Currency(18, 0)
Prediction Date	Prediction_Date__c	Date
Prediction Model	Prediction_Model__c	Picklist
Prediction Name	Name	Text(80)
Stock	Stock__c	Lookup(Stock)

## 2. Relationships

⊕ **Portfolio\_c → Transaction\_c → Stock\_c**

🔗 Many-to-Many via Transaction\_c. One portfolio can hold many stocks, and one stock can belong to many portfolios.

⊕ **Stock\_c → Prediction\_c**

🔗 One-to-Many. A stock can have multiple predictions over time.

⊕ **User (Investor) → Portfolio\_c (One-to-Many).**

## 3. Schema Builder (Visualization)

- **Investor (User) → Portfolio\_c → Transaction\_c → Stock\_c**
- **Stock\_c → Prediction\_c**

📌 This way:

- Admins see all data.
- Investors see **only their portfolios & related transactions.**
- Predictions link back to **stocks** for insights.



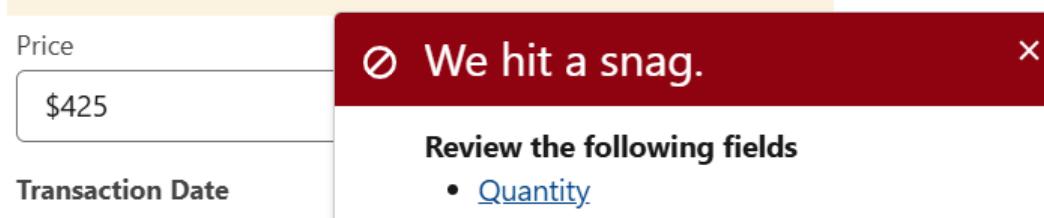
## Phase 4: Process Automation (Admin)

### 1. Validation Rules

- ✓ Prevent bad data entry.
- Transaction\_\_c.Quantity\_\_c > 0
- Formula:  
Quantity\_\_c <= 0  
→ Error Message: “Transaction quantity must be greater than zero.”

The screenshot shows the Salesforce Setup interface with the Object Manager selected. Under the Transaction object, a validation rule named "Quantity\_Check" is displayed. The rule's formula is "Quantity\_\_c <= 0" and its error message is "Transaction quantity must be greater than zero." The rule is active. The sidebar on the left lists various setup options like Details, Fields & Relationships, Page Layouts, etc.

A transaction record is being edited. The "Quantity" field contains "-1". A red error message box appears below the field stating: "Transaction quantity must be greater than zero."



## 2. Flow (Record-Triggered)

⚡ Type: After-Save Flow on Transaction\_\_c

Trigger: When a new transaction is inserted or updated.

Logic:

Get all transactions related to the portfolio.

Calculate :

**SUM(Quantity\_\_c \* Price\_\_c).**

Update :

**Portfolio\_\_c.Total\_Value\_\_c.**

💡 This keeps portfolio valuation auto-updated in real-time.

The screenshot shows the Apex Trigger Detail page for a trigger named 'TransactionTrigger'. The page includes the following details:

Name	Code Coverage	sObject Type
TransactionTrigger	0% (0/28)	Transaction
Created By	Parth Barnote, 9/15/2025, 12:03 AM	Status
Namespace Prefix		Last Modified By
		Parth Barnote, 9/15/2025, 12:03 AM

Below the details, there are tabs for 'Apex Trigger', 'Version Settings', and 'Trace Flags'. The 'Apex Trigger' tab is selected, showing the trigger code:

```
1trigger TransactionTrigger on Transaction__c (after insert, after update, after delete, after undelete) {
2
3    // Step 1: Collect impacted Portfolio Ids
4    Set<Id> portfolioIds = new Set<Id>();
5
6    if(Trigger.isInsert || Trigger.isUpdate || Trigger.isUndelete){
7        for(Transaction__c t : Trigger.new){
8            if(t.Portfolio__c != null){
9                portfolioIds.add(t.Portfolio__c);
10            }
11        }
12    }
13
14    if(Trigger.isUpdate || Trigger.isDelete){
15        for(Transaction__c t : Trigger.old){
16            if(t.Portfolio__c != null){
17                portfolioIds.add(t.Portfolio__c);
18            }
19        }
20    }
21}
```

```

18     }
19   }
20 }
21
22 if(portfoliods.isEmpty()){
23   return;
24 }
25
26 // Step 2: Query related Transactions
27 Map<Id, Decimal> portfolioTotals = new Map<Id, Decimal>();
28 for(Transaction__c txn : [
29   SELECT Id, Portfolio__c, Quantity__c, Price__c
30   FROM Transaction__c
31   WHERE Portfolio__c IN :portfoliods
32 ]){
33   if(!portfolioTotals.containsKey(txn.Portfolio__c)){
34     portfolioTotals.put(txn.Portfolio__c, 0);
35   }
36   Decimal currentTotal = portfolioTotals.get(txn.Portfolio__c);
37   currentTotal += (txn.Quantity__c != null ? txn.Quantity__c : 0) *
38     (txn.Price__c != null ? txn.Price__c : 0);
39   portfolioTotals.put(txn.Portfolio__c, currentTotal);
40 }
41
42 // Step 3: Update Portfolios
43 List<Portfolio__c> portfoliosToUpdate = new List<Portfolio__c>();
44 for(Id pld : portfoliods){
45   Portfolio__c p = new Portfolio__c(Id = pld);
46   p.Total_Value__c = portfolioTotals.containsKey(pld) ? portfolioTotals.get(pld) : 0;
47   portfoliosToUpdate.add(p);
}

```

```

31   WHERE Portfolio__c IN :portfoliods
32 ]){
33   if(!portfolioTotals.containsKey(txn.Portfolio__c)){
34     portfolioTotals.put(txn.Portfolio__c, 0);
35   }
36   Decimal currentTotal = portfolioTotals.get(txn.Portfolio__c);
37   currentTotal += (txn.Quantity__c != null ? txn.Quantity__c : 0) *
38     (txn.Price__c != null ? txn.Price__c : 0);
39   portfolioTotals.put(txn.Portfolio__c, currentTotal);
40 }
41
42 // Step 3: Update Portfolios
43 List<Portfolio__c> portfoliosToUpdate = new List<Portfolio__c>();
44 for(Id pld : portfoliods){
45   Portfolio__c p = new Portfolio__c(Id = pld);
46   p.Total_Value__c = portfolioTotals.containsKey(pld) ? portfolioTotals.get(pld) : 0;
47   portfoliosToUpdate.add(p);
48 }
49
50 if(!portfoliosToUpdate.isEmpty()){
51   update portfoliosToUpdate;
52 }
53}

```

# 📍 Apex Programming (Developer)

## 1.SOQL (Salesforce Object Query Language)

SOQL lets you fetch records from Salesforce objects (like SQL).

Example 1: Get all Stocks in IT Sector

```
1  public class Stock {  
2      public static void demo()  
3      {  
4          List<Stock__c> itStocks = [  
5              SELECT Name, Current_Price__c, Sector__c  
6              FROM Stock__c  
7              WHERE Sector__c = 'IT'  
8          ];  
9          System.debug(itStocks);  
10     }
```

```
13:06:27:036    USER_DEBUG    [9]DEBUG|(Stock__c:{Name=TCS, Current_Price__c=4532, Sector__c=IT, Id=a0AgL000001qbm5UAA})
```

Example 2: Get Predictions for a Particular Stock

```
List<Prediction__c> stockPredictions = [  
SELECT Predicted_Price__c, Confidence_Score__c  
FROM Prediction__c  
WHERE Stock__c = 'a0AgL000001qbm5UAA'  
ORDER BY Predicted_Price__c DESC  
];  
System.debug(stockPredictions);  
}
```

13:06:27:058    USER\_DEBUG    |17||DEBUG|(Prediction\_c:{Predicted\_Price\_c=50000, Confidence\_Score\_c=90, Id=a0DgL00000BkbPdUA})

## 2.Apex Triggers

Triggers allow you to run automation when a record is created, updated, or deleted.

```
1 trigger UpdatePortfolioValue on Transaction_c (after insert, after update) {
2     Set<Id> portfolioIds = new Set<Id>();
3
4     // Collect Portfolio IDs from new transactions
5     for (Transaction_c t : Trigger.new) {
6         portfolioIds.add(t.Portfolio_c);
7     }
8
9     // Fetch related portfolios
10    List<Portfolio_c> portfolios = [
11        SELECT Id, Total_Value_c,
12            (SELECT Quantity_c, Price_c FROM Transactions_r)
13        FROM Portfolio_c
14        WHERE Id IN :portfolioIds
15    ];
16
17    // Recalculate Total Value
18    for (Portfolio_c p : portfolios) {
19        Decimal total = 0;
20        for (Transaction_c t : p.Transactions_r) {
21            total += t.Quantity_c * t.Price_c;
22        }
23        p.Total_Value_c = total;
24    }
25
26    update portfolios;
27 }
28 }
```

- ✓ This trigger ensures whenever a buy/sell transaction happens, the portfolio value updates automatically.