

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).

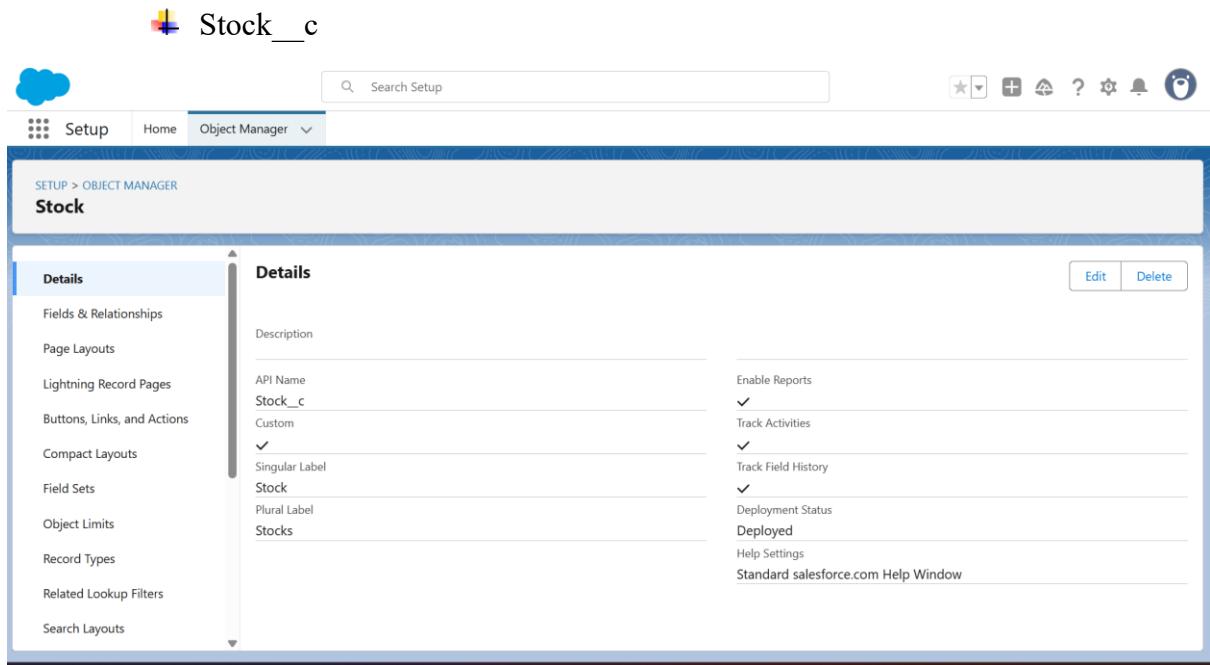
- Deploy to production only after validation.

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

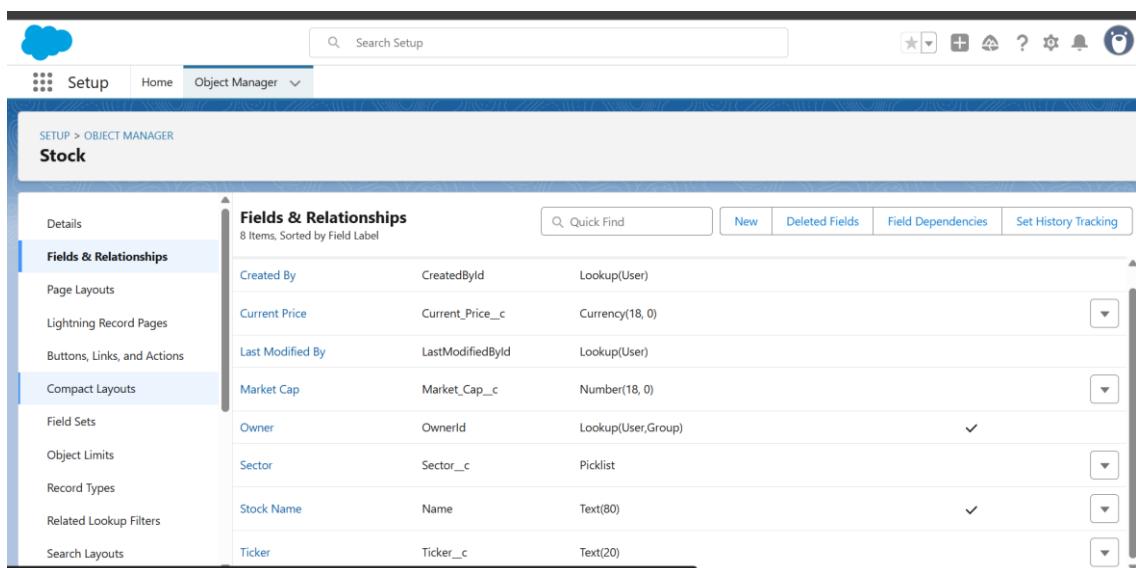
1. Custom Objects & Key Fields



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 Fields & Relationships section for the Stock__c object. The left sidebar is identical to the previous screenshot. The main pane lists the fields and their relationships: 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)), and Ticker (Ticker, Text(20)).

Portfolio_c

Setup > OBJECT MANAGER
Portfolio

Details

Description	<input type="text"/>
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	

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:

- 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)		

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

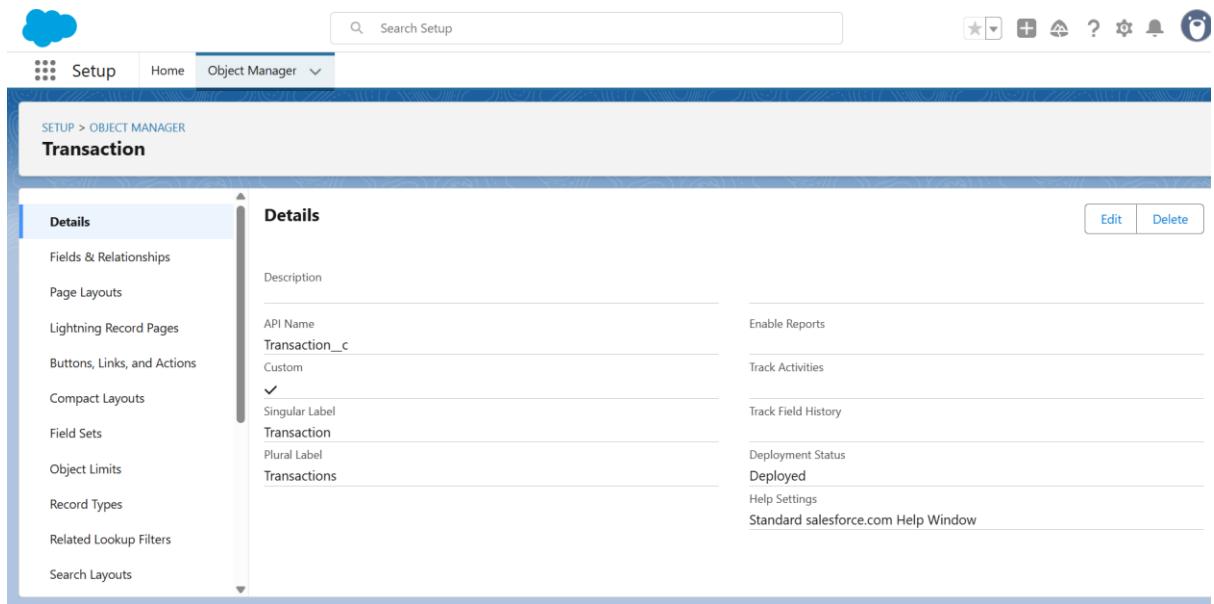
Object Limits

Record Types

Related Lookup Filters

Search Layouts

Transaction__c (Junction between Stock & Portfolio)



Object Manager

Transaction

Details

Description

API Name
Transaction__c

Singular Label
Transaction

Plural Label
Transactions

Enable Reports

Track Activities

Track Field History

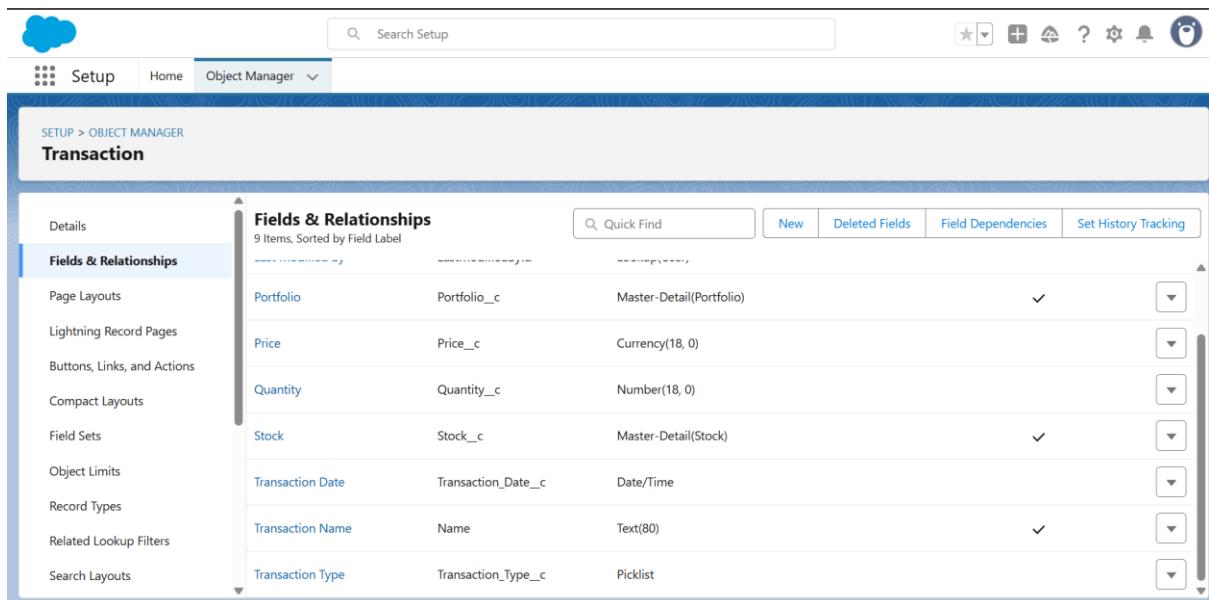
Deployment Status
Deployed

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)



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

SETUP > OBJECT MANAGER
Prediction

Details	
Fields & Relationships	Description
Page Layouts	API Name
Lightning Record Pages	Prediction__c
Buttons, Links, and Actions	Custom
Compact Layouts	✓
Field Sets	Singular Label
Object Limits	Prediction
Record Types	Plural Label
Related Lookup Filters	Predictions
Search Layouts	Enable Reports
	✓
	Track Activities
	✓
	Track Field History
	✓
	Deployment Status
	Deployed
	Help Settings
	Standard salesforce.com Help Window

- 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.)

SETUP > OBJECT MANAGER
Prediction

Fields & Relationships		
9 Items, Sorted by Field Label		
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 for the Transaction object. On the left, a sidebar lists various setup options like Details, Fields & Relationships, Page Layouts, etc. The main area is titled "Transaction Validation Rule" for the "Quantity_Check" rule. It shows the rule name, formula (Quantity__c <= 0), error message ("Transaction quantity must be greater than zero."), and active status. The right side of the screen shows the transaction record with a validation error for the Quantity field.

The screenshot shows a transaction record with fields for Price (\$425) and Transaction Date. The Quantity field has a value of -1, which is highlighted with a red border and an error message: "Transaction quantity must be greater than zero." A modal window titled "We hit a snag." provides instructions to review the Quantity field.

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 Triggers page in the Salesforce Setup. The trigger is named "TransactionTrigger". It is an After-Save trigger on the "Transaction__c" object. The trigger code is as follows:

```
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}

```

