1 NF (the table is already in 1 NF) ha! Trick question

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **UserID** | **UserName** | **StockID** | **StockName** | **StockType** | **StockTypeID** | **Profit/Loss(per share)** |
| 1 | Rakshit | S001 | Adani Energy | Energy | T0011 | 6000 |
| 1 | Rakshit | S002 | TCS | IT | T0031 | -5000 |
| 2 | Sirjan | S002 | TCS | IT | T0031 | 1050 |
| 3 | Kushal | S003 | HDFC | Finance | T0056 | 350 |

Pros :

* Removal of unnecessary repeated value
* No issue while doing CRUD
* Creation of different rows makes it easier to work instead of “,”

Cons:

* Data redundancy happens naturally since there are more rows now with repeated info (not all)
* Wasted space that could be solved in 3NF or Bcnf

2NF

Relations : user id -> username

Stock id -> stock name -> stock type -? Stock type ID

|  |  |  |
| --- | --- | --- |
| **UserID** | **StockID** | **Profit/Loss(per share)** |
| 1 | S001 | 6000 |
| 1 | S002 | -5000 |
| 2 | S002 | 1050 |
| 3 | S003 | 350 |

|  |  |
| --- | --- |
| User id | username |
| 1 | RAKSHIT |
| 2 | SIRJAN |
| 3 | KUSHAL |

|  |  |  |  |
| --- | --- | --- | --- |
| Stock id | Stock name | Stock type | Stock type ID |
| S001 | ADANI ENERGY | ENERGY | T0011 |
| S002 | TCS | IT | T0031 |
| S003 | HDFC | FINANCE | T0056 |

Cons:

* Lot of anomalies can occur for example, insertion, deletion can become a hassle
* For example, inserting stock type could create repeated values

Pros:

* Redundant data reduced more effectively
* Does not have any partial dependancies

3 NF

RELATIONS: user id -> user name

Stock type -> stock type id

Stock id -> stock name

User id, stock id -> profit / loss per share

|  |  |
| --- | --- |
| User id | username |
| 1 | RAKSHIT |
| 2 | SIRJAN |
| 3 | KUSHAL |

|  |  |
| --- | --- |
| Stock type | Stock type ID |
| ENERGY | T0011 |
| IT | T0031 |
| FINANCE | T0056 |

|  |  |
| --- | --- |
| Stock id | Stock name |
| S001 | ADANI ENERGY |
| S002 | TCS |
| S003 | HDFC |

|  |  |  |
| --- | --- | --- |
| **UserID** | **StockID** | **Profit/Loss(per share)** |
| 1 | S001 | 6000 |
| 1 | S002 | -5000 |
| 2 | S002 | 1050 |
| 3 | S003 | 350 |

Pros :

* No transitive dependency on non prime attribute
* Not as ugly as 2NF lol
* Database becomes much smaller
* High data integrity i.e. CRUD easier and more effective operations

Cons:

* Join dependancies and other complex dependancies can be an issue later on
* Can lead to slow performance as now there are much more tables

For Milestone 2:

Here are the assumptions I am making:

1. No intraday trading is allowed
2. If an order is placed, an order for purchasing stocks is successful i.e. there are no rejected trades and likewise, for deletion

Exported image on Git hub

Mile Stone 3)

1. Let date and stock id be given

SELECT SUM((close – open)) \* quantity from stocks

Select SUM(quantity) from orders

WHERE stock id = <given> && date = <given>

1. Let date be given

Select sum((close-open))\*quantity from stocks

From holdings INNER JOIN stock

On holdings .stockId= stock.stock\_id

Where UserID given AND stocks.Date = given;

1. Not able to do this …
2. Let stock id be given and user id be given

Select AVG(purchaseprice \* shares) From holdings

Where StockID <given> && UserId <given>;

Select SUM((Current - BuyingPrice)\*quantity)

From holding\_info INNER JOIN stock\_details

On holding\_info.StockID = stock\_details.StockID

Where date = (Select MAX(date) from stock\_details);

1. Let user id be given

SELECT (today.close-lastmonth.close)/prev.close\*100 as percentage

FROM

(

SELECT close,StockID

FROM stock

Where date = CURDATE() AND StockID IN

(

SELECT StockID, purchaseprice

FROM holdings

)

) as today

INNER JOIN

( SELECT close,StockID

FROM stocks

Where date = CURDATE()- INTERVAL(30) AND StockID IN

(

SELECT StockID

FROM holdings

)

) as lastmonth

ON today.StockID = lastmonth.StockID

WHERE user id given

ORDER BY percentage DESC

LIMIT 1;

(I had another idea for this question… usually in trading, % growth is shown in real time… in fact you can configure that info and check your best performing stock in past day, week, year, or even a custom range... so I would probably have an extra float variable in holdings table where it can give by the % growth and I don’t have to compute it everytime)

