

## MYSQL\_DB

This chapter will only introduce the relationships, dependences, and examples of using the application+user database.

**Table 1: USER\_FULLINFO (contains the full details of an user of our FYP application)**

userid is set to primary key and email is set to be unique value. This is the parent table since there is no foreign key constraint.

```
CREATE TABLE USER_FULLINFO (  
    userid VARCHAR(20) NOT NULL,  
    username VARCHAR(20) NOT NULL,  
    user_password VARCHAR(40) NOT NULL,  
    risk_acceptance_level DOUBLE DEFAULT 0,  
    monthly_expense DOUBLE DEFAULT 0,  
    total_asset DOUBLE DEFAULT 0,  
    principal DOUBLE DEFAULT 0,  
    cash DOUBLE DEFAULT 0,  
    monthly_income DOUBLE DEFAULT NULL,  
    first_invest_day VARCHAR(20) NOT NULL,  
    email VARCHAR(40) NOT NULL,  
    sex VARCHAR(10) NOT NULL,  
    age DOUBLE DEFAULT NULL,  
    PRIMARY KEY (userid),  
    UNIQUE (email)  
);
```

Add dummy data values into the USER\_FULLINFO table

Two users are added, one is Eric, another one is Mary

```
INSERT INTO `USER_FULLINFO` VALUES ('123456789',  
'Eric','ilovehkust',1,2000,31100,28815,100,3000, '01/01/2020','cymaae@connect.ust.hk','M',20);
```

```
INSERT INTO `USER_FULLINFO` VALUES ('324565944', 'Mary', 'ihatehkust',2,1000,1800,1400,300,300,  
'01/03/2020','mary@connect.ust.hk','F',19);
```

Now the current table looks like:

	userid	username	user_password	risk_acceptance_level	monthly_expense	total_asset	principal	cash	monthly_income	first_invest_day	email
▶	123456789	Eric	iovehkust	1	2000	31100	28815	100	3000	01/01/2020	cymaae@connect.ust
	324565944	Mary	ihatehkust	2	1000	1800	1400	300	300	01/03/2020	mary@connect.ust
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

**Table 2: USER\_INFO (contains only brief information of an user, e.g. userid, portfolio\_id (the portfolio owned by the user), and portfolio\_quantity (an user may have more than 1 portfolio))**

userid and portfolio\_id are the joint primary key in this table. Foreign key is userid, that means the userid in USER\_INFO table will reference to the userid in USER\_FULLINFO table. This is the child table of USER\_FULLINFO as the foreign key constraint is set to USER\_FULLINFO.

```
CREATE TABLE USER_INFO (
    userid VARCHAR(20) NOT NULL,
    portfolio_id VARCHAR(20) NOT NULL,
    portfolio_quantity INT DEFAULT 0,
    PRIMARY KEY (userid , portfolio_id),
    FOREIGN KEY (userid)
    REFERENCES USER_FULLINFO (userid)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```

Add dummy data values into the USER\_INFO table

userid = 123456789 has two portfolios (portfolio\_quantity = 2), where the portfolio\_id are 234567891 and 981542267 respectively.

Look back to the USER\_INFO, the username of userid = 123456789 is Eric

```
INSERT INTO `USER_INFO` VALUES ('123456789', '234567891',2);
```

```
INSERT INTO `USER_INFO` VALUES ('123456789', '981542267',2);
```

```
INSERT INTO `USER_INFO` VALUES ('324565944', '785112123',1);
```

Now the current table looks like:

	userid	portfolio_id	portfolio_quantity
▶	123456789	234567891	2
	123456789	981542267	2
	324565944	785112123	1

**Table 3: TRANSACTION\_HISTORY (contains the buy price and sell price, etc the transaction history of users)**

userid , portfolio\_id , stock , begin\_trade\_time are the joint primary key in this table. userid , portfolio\_id are the joint foreign key that references to the table USER\_INFO. That means TRANSACTION\_HISTORY table is the child of USER\_INFO table.

```
CREATE TABLE TRANSACTION_HISTORY (
    userid VARCHAR(20) NOT NULL,
    portfolio_id VARCHAR(20) NOT NULL,
    stock VARCHAR(20) NOT NULL,
    begin_trade_time VARCHAR(20) NOT NULL,
    end_trade_time VARCHAR(20) DEFAULT NULL,
    buy_price DOUBLE NOT NULL,
    sell_price DOUBLE DEFAULT NULL,
    PRIMARY KEY (userid , portfolio_id , stock , begin_trade_time),
    FOREIGN KEY (userid , portfolio_id)
        REFERENCES USER_INFO (userid , portfolio_id)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```

Add dummy data values into the TANSACTION\_HISTOTY table:

```
INSERT INTO `TRANSACTION_HISTORY`
VALUES('123456789','234567891','AAPL','5/6/2021',NULL,460,NULL);
```

```
INSERT INTO `TRANSACTION_HISTORY` VALUES('123456789','234567891','Goldman
Sachs','2/3/2021',NULL,291.5,NULL);
```

```
INSERT INTO `TRANSACTION_HISTORY`
VALUES('123456789','981542267','Google','8/10/2021',NULL,835,NULL);
```

```
INSERT INTO `TRANSACTION_HISTORY`
VALUES('324565944','785112123','AWS','7/15/2021','8/14/2021',35,36.25);
```

Now the current table looks like:

	userid	portfolio_id	stock	begin_trade_time	end_trade_time	buy_price	sell_price
▶	123456789	234567891	AAPL	5/6/2021	NULL	460	NULL
	123456789	234567891	Goldman Sachs	2/3/2021	NULL	291.5	NULL
	123456789	981542267	Google	8/10/2021	NULL	835	NULL
	324565944	785112123	AWS	7/15/2021	8/14/2021	35	36.25

**Table 4: FRIENDSHIP (if has\_friend is true, that means friend\_user\_id will show the friend's userid of users)**

Userid is primary key. Userid is the foreign key that references to USER\_INFO table at the same time. And friend\_user\_id is also the foreign key that references to USER\_INFO table. This friendship table is the child of USER\_INFO.

```
CREATE TABLE FRIENDSHIP (  
    userid VARCHAR(20) NOT NULL,  
    has_friend BOOLEAN,  
    friend_user_id VARCHAR(20) DEFAULT NULL,  
    PRIMARY KEY (userid),  
    FOREIGN KEY (userid)  
        REFERENCES USER_INFO (userid),  
    FOREIGN KEY (friend_user_id)  
        REFERENCES USER_INFO (userid)  
);
```

Add dummy values into FRIENDSHIP table:

```
INSERT INTO `FRIENDSHIP` VALUES ('123456789', true, '324565944');
```

```
INSERT INTO `FRIENDSHIP` VALUES ('324565944', true, '123456789');
```

Now the current table looks like:

	userid	has_friend	friend_user_id
▶	123456789	1	324565944
	324565944	1	123456789

Note that in this example, the friend\_user\_id=324565944 is the friend of an user with userid=123456789

Table 5: PORTFOLIO (contains the details of portfolio owned by users)

portfolio\_id , stock , stock\_action are the joint primary key. userid , portfolio\_id is the foreign key that references to USER\_INFO table. userid , portfolio\_id , stock is the foreign key that references to TRANSACTION\_HISTORY table at the same time. These means PORTFOLIO table is the child of USER\_INFO table and TRANSACTION\_HISTORY table at the same time.

```
CREATE TABLE PORTFOLIO (  
    userid VARCHAR(20) NOT NULL,  
    portfolio_id VARCHAR(20) NOT NULL,  
    stock VARCHAR(20) NOT NULL,  
    profit DOUBLE DEFAULT 0,  
    return_rate DOUBLE DEFAULT 0,  
    dividend_total DOUBLE DEFAULT 0,  
    investment_horizon DOUBLE DEFAULT 0,  
    stock_action VARCHAR(4) NOT NULL, // stock_action: buy/ hold/ sell  
    num_of_share DOUBLE DEFAULT 0,  
    total_balance DOUBLE DEFAULT 0,  
    principal DOUBLE DEFAULT 0,  
    PRIMARY KEY (portfolio_id , stock , stock_action),  
    FOREIGN KEY (userid , portfolio_id)  
        REFERENCES USER_INFO (userid , portfolio_id),  
    FOREIGN KEY (userid , portfolio_id , stock)  
        REFERENCES TRANSACTION_HISTORY (userid , portfolio_id , stock)  
    ON DELETE CASCADE ON UPDATE CASCADE  
);  
  
INSERT INTO `PORTFOLIO`  
VALUES('123456789','234567891','AAPL',500,8.6957,300,30,'buy',20,10000,9200);  
  
INSERT INTO `PORTFOLIO` VALUES('123456789','234567891','Goldman  
Sachs',15,2.9160,70,30,'hold',10,3000,2915);  
  
INSERT INTO `PORTFOLIO`  
VALUES('123456789','981542267','Google',800,7.78,500,30,'buy',20,18000,16700);  
  
INSERT INTO `PORTFOLIO` VALUES('324565944','785112123','AWS',0,0,0,30,'buy',40,1500,1400);  
  
INSERT INTO `PORTFOLIO`  
VALUES('324565944','785112123','AWS',50,7.1429,50,30,'sell',40,1500,1400);
```

Now current table looks like:

	userid	portfolio_id	stock	profit	return_rate	dividend_total	investment_horizon	stock_action	num_of_share	total_balance	principal
▶	123456789	234567891	AAPL	500	8.6957	300	30	buy	20	10000	9200
	123456789	234567891	Goldman Sachs	15	2.916	70	30	hold	10	3000	2915
	324565944	785112123	AWS	0	0	0	30	buy	40	1500	1400
	324565944	785112123	AWS	50	7.1429	50	30	sell	40	1500	1400
	123456789	981542267	Google	800	7.78	500	30	buy	20	18000	16700

Table 6: STOCK (contains the details of stocks that are involved in the portfolios)

dates , stock , portfolio\_id , stock\_action are the joint primary key. portfolio\_id , stock , stock\_action are the joint foreign key that references to PORTFOLIO table. This STOCK table is the child of PORTFOLIO table.

```
CREATE TABLE STOCK (
    dates VARCHAR(20) NOT NULL,
    portfolio_id VARCHAR(20) NOT NULL,
    stock VARCHAR(20) NOT NULL,
    industry VARCHAR(40) NOT NULL,
    current_price DOUBLE NOT NULL,
    dividend_per_share DOUBLE DEFAULT 0,
    volatility DOUBLE,
    volume DOUBLE DEFAULT 0,
    PX_OPEN DOUBLE,
    PX_HIGH DOUBLE,
    PX_LOW DOUBLE,
    PX_CLOSE DOUBLE,
    stock_action VARCHAR(4) NOT NULL,
    PRIMARY KEY (dates , stock , portfolio_id , stock_action),
    FOREIGN KEY (portfolio_id , stock , stock_action)
        REFERENCES PORTFOLIO (portfolio_id , stock , stock_action)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

Add dummy values into STOCK table:

```
INSERT INTO `STOCK`
VALUES('5/6/2021','234567891','AAPL','technology',460,15,3.6,5000000,460,480,445,463,'buy');

INSERT INTO `STOCK` VALUES('2/3/2021','234567891','Goldman
Sachs','finance/banking',291.5,7,2.5,900000,291.5,296,290,292,'hold');

INSERT INTO `STOCK`
VALUES('8/10/2021','981542267','Google','technology',835,25,3.1,8000000,835,840,832,835.5,'buy')
;

INSERT INTO `STOCK`
VALUES('7/15/2021','785112123','AWS','technology',35,1.25,1.4,6500000,35,35.8,34,34.6,'buy');

INSERT INTO `STOCK`
VALUES('8/14/2021','785112123','AWS','technology',36.5,1.25,1.25,6500000,36,36.5,35.7,36.4,'sell')
;
```

Now current table looks like:

dates	portfolio_id	stock	industry	current_price	dividend_per_share	volatility	volume	PX_OPEN	PX_HIGH	PX_LOW	PX_CLOSE	stock_action
2/3/2021	234567891	Goldman Sachs	finance/banking	291.5	7	2.5	900000	291.5	296	290	292	hold
5/6/2021	234567891	AAPL	technology	460	15	3.6	5000000	460	480	445	463	buy
7/15/2021	785112123	AWS	technology	35	1.25	1.4	6500000	35	35.8	34	34.6	buy
8/10/2021	981542267	Google	technology	835	25	3.1	8000000	835	840	832	835.5	buy
8/14/2021	785112123	AWS	technology	36.5	1.25	1.25	6500000	36	36.5	35.7	36.4	sell

**Example 1: get the user full details by joining USER\_INFO table and USER\_FULLINFO table with joining key userid**

```
SELECT USER_INFO.userid, USER_INFO.portfolio_id,
USER_FULLINFO.username, USER_FULLINFO.total_asset, USER_FULLINFO.cash,
USER_FULLINFO.email
FROM USER_INFO
LEFT JOIN USER_FULLINFO ON USER_INFO.userid = USER_FULLINFO.userid;
```

Result:

	userid	portfolio_id	username	total_asset	cash	email
▶	123456789	234567891	Eric	31100	100	cymaae@connect.ust.hk
	123456789	981542267	Eric	31100	100	cymaae@connect.ust.hk
	324565944	785112123	Mary	1800	300	mary@connect.ust.hk

**Example 2: get all the transaction records of user with userid = 123456789 by joining USER\_INFO and TRANSACTION\_HISTORY with userid and portfolio\_id as keys**

```
SELECT *
FROM USER_INFO
INNER JOIN TRANSACTION_HISTORY ON
USER_INFO.userid = TRANSACTION_HISTORY.userid
and USER_INFO.portfolio_id = TRANSACTION_HISTORY.portfolio_id
and USER_INFO.userid = '123456789';
```

**Example 3: get the friend id for each user**

```
SELECT distinct USER_INFO.userid, FRIENDSHIP.friend_user_id
FROM USER_INFO
LEFT JOIN FRIENDSHIP ON
USER_INFO.userid = FRIENDSHIP.userid;
```

Result:

	userid	friend_user_id
▶	123456789	324565944
	324565944	123456789

**Example 4: get the portfolio details for each user and show the full details of users at the same time**

```
SELECT USER_INFO.userid, USER_INFO.portfolio_id,
USER_FULLINFO.username, USER_FULLINFO.total_asset, USER_FULLINFO.email,
PORTFOLIO.stock, PORTFOLIO.profit, PORTFOLIO.dividend_total, PORTFOLIO.num_of_share,
PORTFOLIO.principal, PORTFOLIO.stock_action, PORTFOLIO.total_balance
FROM USER_INFO
LEFT JOIN PORTFOLIO ON USER_INFO.userid = PORTFOLIO.userid and USER_INFO.portfolio_id =
PORTFOLIO.portfolio_id
LEFT JOIN USER_FULLINFO ON USER_INFO.userid = USER_FULLINFO.userid
where USER_INFO.userid = '123456789';
```

Result:

userid	portfolio_id	username	total_asset	email	stock	profit	dividend_total	num_of_share	principal	stock_action	total_balance
123456789	234567891	Eric	31100	cymaae@connect.ust.hk	AAPL	500	300	20	9200	buy	10000
123456789	234567891	Eric	31100	cymaae@connect.ust.hk	Goldman Sachs	15	70	10	2915	hold	3000
123456789	981542267	Eric	31100	cymaae@connect.ust.hk	Google	800	500	20	16700	buy	18000



Note that we can then use this query result with python to calculate the stock weighting in a portfolio by utilizing the principal data field. For example, the portfolio 234567891 owned by Eric, two stocks are present. AAPL stock has weighting =  $9200 / (9200 + 2915) = 75.94\%$ , whereas Goldman Sachs stock has weighting =  $2915 / (9200 + 2915) = 24.06\%$

#### Example 5: get the transaction history for each portfolio

```
SELECT PORTFOLIO.userid, PORTFOLIO.portfolio_id, PORTFOLIO.stock, PORTFOLIO.stock_action,  
TRANSACTION_HISTORY.begin_trade_time, TRANSACTION_HISTORY.end_trade_time,  
TRANSACTION_HISTORY.buy_price, TRANSACTION_HISTORY.sell_price  
FROM PORTFOLIO  
LEFT JOIN TRANSACTION_HISTORY ON  
PORTFOLIO.userid = TRANSACTION_HISTORY.userid  
and PORTFOLIO.portfolio_id = TRANSACTION_HISTORY.portfolio_id  
and PORTFOLIO.stock = TRANSACTION_HISTORY.stock;
```

Result:

	userid	portfolio_id	stock	stock_action	begin_trade_time	end_trade_time	buy_price	sell_price
▶	123456789	234567891	AAPL	buy	5/6/2021	NULL	460	NULL
	123456789	234567891	Goldman Sachs	hold	2/3/2021	NULL	291.5	NULL
	123456789	981542267	Google	buy	8/10/2021	NULL	835	NULL
	324565944	785112123	AWS	buy	7/15/2021	8/14/2021	35	36.25
	324565944	785112123	AWS	sell	7/15/2021	8/14/2021	35	36.25

Note that some end\_trade\_time and sell\_price fields have NULL values, this is because the stock\_action of that stock is only either buy/hold but not sell.

### Example 6: get the stock details involved in the portfolios

```
SELECT PORTFOLIO.portfolio_id, PORTFOLIO.stock, PORTFOLIO.stock_action,  
STOCK.dates, STOCK.industry, STOCK.current_price, STOCK.dividend_per_share, STOCK.volatility  
FROM PORTFOLIO  
LEFT JOIN STOCK ON  
PORTFOLIO.stock = STOCK.stock  
and PORTFOLIO.portfolio_id = STOCK.portfolio_id  
and PORTFOLIO.stock_action = STOCK.stock_action;
```

Result:

	portfolio_id	stock	stock_action	dates	industry	current_price	dividend_per_share	volatility
▶	234567891	AAPL	buy	5/6/2021	technology	460	15	3.6
	234567891	Goldman Sachs	hold	2/3/2021	finance/banking	291.5	7	2.5
	981542267	Google	buy	8/10/2021	technology	835	25	3.1
	785112123	AWS	buy	7/15/2021	technology	35	1.25	1.4
	785112123	AWS	sell	8/14/2021	technology	36.5	1.25	1.25

Note that the 'dates' field is showing the date of stock\_action.