Some Code Samples

Stock Buy

- lets say you buy 100 shares of tata steel worth Rs 50.
- We first check if you have at least 5000 in you account
- we deduct 5000 from you account.
- we add 100 shares of tata steel in customer's account. if you already have tata steel share then we avg out the price and update the entry.
- An entry in recent transaction is also added

```
func (c *DbConn) BuyStocks(input models.BuyStocksModel) error {
    //ctx := context.Background()

    today := time.Now().Format("2006-01-02") // today date to sql date
    todayIso := time.Now().Format(time.RFC3339)
    transactionId := ksuid.New() // a time sorted uuid
    stockBuyId := ksuid.New()

    desc := fmt.Sprintf("Bought %d stocks of %s ", input.Quantity,
input.CompanyName)

    /*
        1 stockid
        2 date
        3 customerid
        4 quantity
        5 transaction identity
        6 desc
        7 stockbuyid
    */

    result, err := c.pool.Exec(context.Background(), "with sp as ( select
price from stocks s where s.stock_id = $1),\nbalance as ( select (select
balance from customers where customer_id = $3) > sp.price* $4 as
enough_balance, sp.price*$4 as amount,price from sp),\ncu as (update
customers set balance = balance - sp.price*$4 from sp where customer_id =
$3 and (select enough_balance from balance) = true),\nit as (INSERT INTO
transactions (id, amount, \"date\", customer_id, \"desc\") select $5,
amount, $8, $3, $6 from balance where enough_balance = true),\nisbuy as
(INSERT INTO stocks_buy (id, stock_id, quantity, buy_date, price,
remaining_quantity,customer_id) select $7, $1, $4, $2, price, $4,$3 from
balance where enough_balance = true)\nINSERT INTO customer_stocks as cs
(customer_id, stock_id, quantity, avg_price) select $3, $1, $4, price from
balance where enough_balance = true ON CONFLICT (customer_id,stock_id) DO
update set quantity = cs.quantity + $4, avg_price =
(cs.quantity*cs.avg_price + $4* (select price from balance))/(cs.quantity+
```

```
$4) where cs.stock_id = $1 and cs.customer_id= $3 and (select
enough_balance from balance) = true;", input.StockId, today,
input.CustomerId, input.Quantity, transactionId, desc, stockBuyId,
todayIso)
    fmt.Println("printing err ", err)

    fmt.Println("printing result ", result)

    return err

}
```

- similarly we have SellStocks, BuyMF , SellMF code.

Mutual Fund Sell Trigger

- whenever a mutual fund is sold, we calculate long term and short term gains.
- Calculation is done as per FIFO cost basis method
- There is similar database trigger for stock sell

```
CREATE OR REPLACE FUNCTION public.calculate_tax_liability_mf()
 RETURNS trigger
 LANGUAGE plpgsql
AS $function$
    BEGIN

 with r as (select
        id,customer_id,mf_id,price as buy_price,
        sum(remaining_quantity) over( order by buy_date)
sum_qoh,remaining_quantity oldrem, buy_date, case  when date_part('year',
buy_date::date) = date_part('year', current_date) then 's' else 'l' end
st_lt from mf_buy where remaining_quantity != 0 and customer_id =
new.customer_id and mf_id = new.mf_id),
  x as (update mf_buy m1 set remaining_quantity =  greatest(r.sum_qoh-
new.quantity,0) from r where m1.id = r.id and r.sum_qoh -
m1.remaining_quantity  < new.quantity returning r.oldrem -
m1.remaining_quantity as sold_quantity, r.st_lt,r.buy_price),
 y as ( select sum(sold_quantity * (new.price - buy_price))  total, st_lt
from x group by st_lt)
 insert into tax_liability as tl
values(new.customer_id,date_part('year',current_date), coalesce((select
total from y where y.st_lt='l'),0) , coalesce((select total from y where
y.st_lt = 's'),0))
on conflict (customer_id,year) do update set ltcg = tl.ltcg+
coalesce((select total from y where y.st_lt='l'),0), stcg = tl.stcg +
coalesce((select total from y where y.st_lt = 's'),0) where tl.customer_id
= new.customer_id and tl.year = date_part('year',current_date);
RETURN new;
    END;
```

```
$function$
;
```

Cron Job

- A cron job to daily update the stock and mutual fund prices

```
const { DateTime, Settings } = require("luxon");
const { Pool, Client } = require("pg");
const { randomPricePercentageChange, change } = require("./utils");
var cron = require("node-cron");
const axios = require("axios");
const schedule = require("node-schedule");
const dotenv = require("dotenv");
dotenv.config();
let rule = new schedule.RecurrenceRule();
Settings.defaultZoneName = "Asia/Kolkata";

// your timezone
rule.tz = "Asia/Kolkata";

// runs at 15:00:00
rule.second = 0;
rule.minute = process.env.minute;
rule.hour = process.env.hour;
console.log({ rule });
schedule.scheduleJob(rule, async () => {
  await job();
});
async function job() {
  const { user, host, port, database, password, healthChecksBaseURL } =
    process.env;

  try {
    console.log("inside job");
    let dateBeforeToGenerate = DateTime.fromSQL(
      process.env.dateBeforeToGenerate
    );

    if (!dateBeforeToGenerate.isValid) {
      dateBeforeToGenerate = DateTime.now();
    }
    console.log({ dateBeforeToGenerate });

    const p1 = axios.default.get(`${healthChecksBaseURL}/start`);

    const client = new Client({
      user,
      host,
      port,
      database,
```

```
      password,
    });

    await client.connect();

    const tomorrow = dateBeforeToGenerate.plus({ days: 1 });
    const aYearBefore = tomorrow.plus({ year: -1 });
    console.log({ tomorrow, aYearBefore });
    console.log();
    const result = await client.query(
      `select stock_id,price, (select price from stocks_daily_price sdpold
where "date" = '${aYearBefore.toSQLDate()}' and stock_id = s.stock_id)
year_old_price from stocks s`
    );
    console.log({ result });
    let sq = "with ss as (SELECT * FROM (VALUES";
    const sr = result.rows;
    sr.forEach((row, i) => {
      const newPrice = randomPricePercentageChange(row.price);
      const dailyChange = change(newPrice, row.price);
      const yearlyChange = change(newPrice, row.year_old_price);
      sq += `('${
        row.stock_id
      }', '${tomorrow.toSQLDate()}'::date, ${newPrice}, ${dailyChange},
${yearlyChange})`;

      if (i < sr.length - 1) {
        sq += ",";
      }
    });
    sq += `) AS t (stock_id,date,price,daily_change,yearly_change)), ist
as ( INSERT INTO stocks_daily_price (stock_id, "date", price)  select
stock_id, "date", price from ss) update stocks s set price =
sstable.price, daily_change = sstable.daily_change, yearly_change =
sstable.yearly_change  from ss sstable where s.stock_id = sstable.stock_id
`;
    console.log({ sq });

    const sqResult = await client.query(sq);
    console.log({ sqResult });

    console.log("generating mf data");
    const mfdaily = await client.query(
      `select mf_id,price, (select price from mf_daily_price where "date"
= '${aYearBefore.toSQLDate()}' and mf_id = mf.mf_id) year_old_price from
mutual_funds mf`
    );
    let mfsq = "with mfs as (SELECT * FROM (VALUES";
    const mfr = mfdaily.rows;
    mfr.forEach((row, i) => {
      const newPrice = randomPricePercentageChange(row.price);
      const dailyChange = change(newPrice, row.price);
      const yearlyChange = change(newPrice, row.year_old_price);
      mfsq += `('${
```

```
        row.mf_id
      }', '${tomorrow.toSQLDate()}'::date, ${newPrice}, ${dailyChange},
${yearlyChange})`;

      if (i < mfr.length - 1) {
        mfsq += ",";
      }
    });
    mfsq += `) AS t (mf_id,date,price,daily_change,yearly_change)), ist as
( INSERT INTO mf_daily_price (mf_id, "date", price)  select mf_id, "date",
price from mfs) update mutual_funds mf set price = mfstable.price,
daily_change = mfstable.daily_change, yearly_change =
mfstable.yearly_change  from mfs mfstable where mf.mf_id = mfstable.mf_id
`;
    // console.log({sq})
    console.log({ mfsq });

    const mfsqResult = await client.query(mfsq);
    // mfsqResult.rowCount
    console.log({ mfsqResult });
    await client.end();
    const p2 = axios.default.get(`${healthChecksBaseURL}`);
    await Promise.all([p1, p2]);
  } catch (err) {
    console.log("error occured", err);
    await axios.default.get(`${healthChecksBaseURL}/fail`).catch((err) =>
{
      console.log("healthcheck fail endpoint error", err);
    });
  }
}

// job();
```