

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

SC2207 Lab 5 Team 1

Liong Xun Qi (U2322609H)

Glynis Looi Xin Lin (U2321198L)



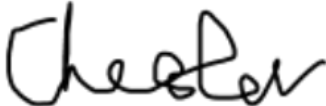


Balajee Viswanath Akshay Narayanan (U2323942B)

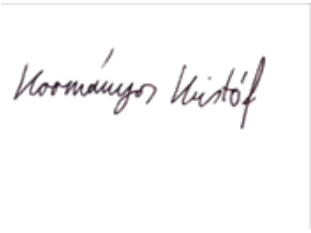
Chester Chan Hong Kai (U2321708L)

Kristof Kormanyos (N2401546A)

Jothilingam Dheeraj (U2321317H)

Individual Contribution Form:

Full Name	Individual Contribution to Lab5 Submission	Percentage of Contribution	Signature
Liong Xun Qi	Create & refine SQL queries. Cross check queries with Schema. Fill in details in the report.	16.67%	
Balajee Viswanath Akshay Narayanan	Create & refine SQL queries. Cross check queries with Schema. Fill in details in the report.	16.67%	
Chester Chan Hong Kai	Create & refine SQL queries. Cross check queries with Schema. Fill in details in the report.	16.67%	
Glynis Looi Xin Lin	Create & refine SQL queries. Cross check queries with Schema. Fill in details in the report.	16.67%	
Jothilingam Dheeraj	Create & refine SQL queries. Cross check queries with Schema. Fill in details in the report.	16.67%	

Kristof Kormanyos	Create & refine SQL queries. Cross check queries with Schema. Fill in details in the report.	16.67%	
-------------------	--	--------	---

To include:

- SQL DDL commands for table creation (from Lab 4).
- SQL statements to solve the queries in Appendix B and additional queries (if any). Each query should be immediately followed by the query output. Briefly explain how the output is obtained.
- A printout of all table records.
- Description of any additional effort made.

SQL DDL Commands for Table Creation

```
use giveAPlease;
```

```
-- Reset : Order in reverse as creation to eradicate any errors
DROP TABLE IF EXISTS TRANSACTION_FEES;
DROP TABLE IF EXISTS TRANSACTION1;
DROP TABLE IF EXISTS UNREALIZED_GAIN_LOSS;
DROP TABLE IF EXISTS INVESTED_VALUE;
DROP TABLE IF EXISTS FUND_IN_PORTFOLIO;
DROP TABLE IF EXISTS BOND_IN_PORTFOLIO;
DROP TABLE IF EXISTS STOCK_IN_PORTFOLIO;
DROP TABLE IF EXISTS R3TURNS;
DROP TABLE IF EXISTS PORTFOLIO1;
DROP TABLE IF EXISTS FUND;
DROP TABLE IF EXISTS BOND;
DROP TABLE IF EXISTS STOCK;
DROP TABLE IF EXISTS ASSET;
DROP TABLE IF EXISTS FINANCIAL_GOAL;
DROP TABLE IF EXISTS RISK_TOLERANCE;
DROP TABLE IF EXISTS INVESTOR;
```

```

-- Create INVESTOR Table
CREATE TABLE INVESTOR (
    Phone VARCHAR(15) PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Gender CHAR(1),
    DoB DATE,
    AnnualIncome DECIMAL(15, 2),
    Company VARCHAR(100)
);

-- Create RISK_TOLERANCE Table
CREATE TABLE RISK_TOLERANCE (
    Phone VARCHAR(15),
    RiskLevel VARCHAR(20),
    Q1A VARCHAR(255),
    Q2A VARCHAR(255),
    Q3A VARCHAR(255),
    Q4A VARCHAR(255),
    Q5A VARCHAR(255),
    PRIMARY KEY (Phone, RiskLevel),
    FOREIGN KEY (Phone) REFERENCES INVESTOR(Phone)
);

-- Create FINANCIAL_GOAL Table
CREATE TABLE FINANCIAL_GOAL (
    Goal VARCHAR(100),
    Phone VARCHAR(15),
    Amount DECIMAL(15, 2),
    Timeline INT,
    PRIMARY KEY (Goal, Phone),
    FOREIGN KEY (Phone) REFERENCES INVESTOR(Phone)
);

-- Create ASSET Table
CREATE TABLE ASSET (
    AssetID VARCHAR(20) PRIMARY KEY,
    Name VARCHAR(100),
    Price DECIMAL(15, 2)
);

-- Create STOCK Table
CREATE TABLE STOCK (
    AssetID VARCHAR(20) PRIMARY KEY,
    P_ERatio DECIMAL(10, 2),
    EPS DECIMAL(10, 2),
    EBITDA DECIMAL(15, 2),
    FOREIGN KEY (AssetID) REFERENCES ASSET(AssetID)
);

-- Create BOND Table

```

```

CREATE TABLE BOND (
  AssetID VARCHAR(20) PRIMARY KEY,
  InterestRate DECIMAL(5, 2),
  MaturityDate DATE,
  FOREIGN KEY (AssetID) REFERENCES ASSET(AssetID)
);

-- Create FUND table
CREATE TABLE FUND (
  AssetID VARCHAR(20) PRIMARY KEY,
  ExpenseRatio DECIMAL(5, 2),
  DividendYield DECIMAL(5, 2),
  FOREIGN KEY (AssetID) REFERENCES ASSET(AssetID)
);

-- Create PORTFOLIO Table (Decomposed into PORTFOLIO1 and ReTURNS)
CREATE TABLE PORTFOLIO1 (
  Phone VARCHAR(15),
  PID VARCHAR(20),
  MarketValue DECIMAL(15, 2),
  InceptionDate DATE,
  Fee DECIMAL(15, 2),
  AnnualizedReturn DECIMAL(5, 2),
  PRIMARY KEY (PID, Phone),
  FOREIGN KEY (Phone) REFERENCES INVESTOR(Phone)
);

-- Create R3TURNS Table
CREATE TABLE R3TURNS (
  MarketValue DECIMAL(15, 2),
  InceptionDate DATE,
  AnnualizedReturn DECIMAL(5, 2),
  PRIMARY KEY (MarketValue, InceptionDate)
);

-- Create STOCK_IN_PORTFOLIO Table
CREATE TABLE STOCK_IN_PORTFOLIO (
  ID VARCHAR(20) PRIMARY KEY,
  PID VARCHAR(20),
  Phone VARCHAR(15),
  StartDate DATE,
  AllocationRatio DECIMAL(5, 2),
  PostTradeCO VARCHAR(100),
  AssetID VARCHAR(20),
  FOREIGN KEY (AssetID) REFERENCES STOCK(AssetID),
  FOREIGN KEY (PID, Phone) REFERENCES PORTFOLIO1(PID, Phone)
);

-- Create BOND_IN_PORTFOLIO Table
CREATE TABLE BOND_IN_PORTFOLIO (

```

```

ID VARCHAR(20) PRIMARY KEY,
PID VARCHAR(20),
Phone VARCHAR(15),
StartDate DATE,
AllocationRatio DECIMAL(5, 2),
PostTradeCO VARCHAR(100),
AssetID VARCHAR(20),
FOREIGN KEY (AssetID) REFERENCES BOND(AssetID),
FOREIGN KEY (PID, Phone) REFERENCES PORTFOLIO1(PID, Phone)
);

```

```

-- Create FUND_IN_PORTFOLIO Table
CREATE TABLE FUND_IN_PORTFOLIO (
  ID VARCHAR(20) PRIMARY KEY,
  PID VARCHAR(20),
  Phone VARCHAR(15),
  StartDate DATE,
  AllocationRatio DECIMAL(5, 2),
  PostTradeCO VARCHAR(100),
  AssetID VARCHAR(20),
  FOREIGN KEY (AssetID) REFERENCES FUND(AssetID),
  FOREIGN KEY (PID, Phone) REFERENCES PORTFOLIO1(PID, Phone)
);

```

```

-- Create INVESTED_VALUE Table
CREATE TABLE INVESTED_VALUE (
  Phone VARCHAR(15),
  PID VARCHAR(20),
  Date DATE,
  Amount DECIMAL(15, 2),
  PRIMARY KEY (Phone, PID, Date),
  FOREIGN KEY (PID, Phone) REFERENCES PORTFOLIO1(PID, Phone)
);

```

```

-- Create UNREALIZED_GAIN_LOSS Table
CREATE TABLE UNREALIZED_GAIN_LOSS (
  Phone VARCHAR(15) NOT NULL,
  PID VARCHAR(20) NOT NULL,
  Date DATE NOT NULL,
  Amount DECIMAL(15, 2),
  PRIMARY KEY (Phone, PID, Date),
  FOREIGN KEY (PID, Phone) REFERENCES PORTFOLIO1(PID, Phone)
);

```

```

-- Create TRANSACTION Table (Decompose into TRANSACTION1 and
TRANSACTION_FEES)
CREATE TABLE TRANSACTION1 (
  ID VARCHAR(20),
  Date DATE,
  PID VARCHAR(20),

```

```

Phone VARCHAR(15),
Type VARCHAR(50),
PRIMARY KEY (ID, Date),
FOREIGN KEY (PID, Phone) REFERENCES PORTFOLIO1(PID, Phone)
);

-- Create TRANSACTION_FEES Table
CREATE TABLE TRANSACTION_FEES (
    Type VARCHAR(50) PRIMARY KEY,
    Fee DECIMAL(5, 2)
);

```

Table Records

INVESTOR

	Phone	Name	Gender	DoB	AnnualIncome	Company
1	81510001	Sherlock Holmes	M	1985-04-12	85000.00	DBS
2	81510002	Harry Potter	M	1990-08-25	92000.00	OCBC
3	81510003	James Bond	M	1990-03-02	92000.00	OCBC
4	81510004	Jack Sparrow	M	1990-08-26	92000.00	OCBC
5	81510005	Jay Gatsby	M	1990-08-22	92000.00	OCBC
6	81510006	Grey Gandalf	M	1990-08-25	92000.00	DBS
7	81510007	Victor Frankenstein	M	1990-08-25	92000.00	DBS
8	81510008	Louis Litt	F	2000-08-25	92000.00	Bosch
9	81510009	John Reacher	F	2001-12-01	110000.00	Bosch
10	81510010	Arya Stark	M	1999-01-14	110000.00	Bosch

RISK_TOLERANCE

	Phone	RiskLevel	Q1A	Q2A	Q3A	Q4A	Q5A
1	81510001	Conservative	A	B	A	A	A
2	81510001	Moderate	A	B	A	B	B
3	81510002	Aggressive	C	C	C	B	C
4	81510003	Conservative	A	A	A	C	A
5	81510004	Moderate	A	B	B	A	A
6	81510005	Aggressive	C	C	C	A	B
7	81510005	Moderate	C	B	B	B	C
8	81510006	Conservative	A	A	A	B	A
9	81510007	Moderate	B	B	A	A	B
10	81510008	Conservative	A	A	B	B	A
11	81510009	Aggressive	C	C	C	C	C
12	81510009	Moderate	B	B	C	A	B
13	81510010	Moderate	B	B	A	A	C

FINANCIAL_GOAL

	Goal	Phone	Amount	Timeline
1	Child Education	81510003	200000.00	20
2	Child Education	81510004	200000.00	15
3	Child Education	81510009	200000.00	15
4	House Purchase	81510001	300000.00	20
5	House Purchase	81510005	300000.00	15
6	Lambo Purchase	81510002	300000.00	7
7	Mansion Building	81510009	300000.00	5
8	Property Purchase	81510006	300000.00	25
9	Retirement	81510001	500000.00	30
10	Retirement	81510004	500000.00	20
11	Retirement	81510007	500000.00	17
12	Retirement	81510008	500000.00	35
13	Rolex Collection	81510005	200000.00	5
14	Support Fund	81510010	500000.00	13

ASSET

	AssetID	Name	Price
1	BND001	iShares 1-3 Year Treasury Bond ETF (SHY)	150.75
2	BND111	Vanguard Tax-Exempt Bond ETF (VTEB)	101.50
3	BND170	Vanguard Total Bond Market ETF (BND)	95.25
4	BND189	iShares 20+ Year Treasury Bond ETF (TLT)	102.50
5	BND222	Schwab US TIPS ETF (SCHP)	98.75
6	BND333	SPDR Portfolio Mortgage-Backed Bond ETF (SPMB)	102.20
7	BND444	iShares Core US Aggregate Bond ETF (AGG)	99.10
8	BND555	Nikko AM SGD Investment Grade Corporate Bond ET...	100.80
9	BND666	Vanguard Short-Term Corporate Bond ETF (VCSH)	150.75
10	BND777	iShares JP Morgan USD Asia Credit Bond Index ETF (...)	97.90
11	FND010	Global X Lithium & Battery Tech ETF	132.80
12	FND111	Schwab U.S. Broad Market ETF	110.40
13	FND222	ARK Innovation ETF	135.20
14	FND333	iShares Core MSCI EAFE ETF	115.80
15	FND444	Fidelity® Real Estate Income ETF	140.60
16	FND555	Vanguard Total Stock Market ETF	112.90
17	FND666	Invesco QQQ Trust	130.10
18	FND777	iShares Core S&P Mid-Cap ETF	118.30
19	FND888	SPDR® Portfolio Aggregate Bond ETF	145.70
20	FND999	iShares Russell 2000 ETF	117.50
21	STK010	The Home Depot, Inc.	330.60
22	STK101	JPMorgan Chase & Co.	160.25
23	STK202	UnitedHealth Group Incorporated	520.80
24	STK303	The Procter & Gamble Company	155.50
25	STK404	The Coca-Cola Company	60.10
26	STK505	Alphabet Inc.	2750...
27	STK606	Meta Platforms, Inc.	350.75
28	STK707	Visa Inc.	240.30
29	STK808	Walmart Inc.	165.90
30	STK909	Pfizer Inc.	35.45

STOCK

	AssetID	P_ERatio	EPS	EBITDA
1	STK010	32.80	6.90	19000000000.00
2	STK101	22.50	4.80	15000000000.00
3	STK202	28.30	6.10	18000000000.00
4	STK303	25.00	5.20	12000000000.00
5	STK404	20.70	3.90	10000000000.00
6	STK505	35.60	7.50	20000000000.00
7	STK606	40.20	8.20	25000000000.00
8	STK707	18.90	4.30	9000000000.00
9	STK808	23.40	5.00	11000000000.00
10	STK909	27.10	5.80	14000000000.00

BOND

	AssetID	InterestRate	MaturityDate
1	BND001	4.25	2026-12-31
2	BND111	4.70	2030-12-15
3	BND170	4.00	2030-09-30
4	BND189	3.75	2045-03-15
5	BND222	4.10	2033-06-30
6	BND333	4.30	2029-09-30
7	BND444	4.00	2031-03-31
8	BND555	4.60	2028-11-30
9	BND666	4.50	2028-06-30
10	BND777	4.20	2034-04-30

FUND

	AssetID	ExpenseRatio	DividendYield
1	FND010	0.65	3.60
2	FND111	0.15	2.00
3	FND222	0.75	3.50
4	FND333	0.20	1.80
5	FND444	0.80	4.00
6	FND555	0.18	2.20
7	FND666	0.70	3.80
8	FND777	0.22	1.90
9	FND888	0.85	4.20
10	FND999	0.25	2.10

PORTFOLIO1

	Phone	PID	MarketValue	InceptionDate	Fee	AnnualizedReturn
1	81510001	P001	95000.00	2020-03-15	1.30	5.20
2	81510001	P002	105000.00	2021-06-20	1.30	6.00
3	81510002	P003	150000.00	2019-09-01	1.30	9.80
4	81510003	P004	88000.00	2022-01-10	1.30	4.80
5	81510004	P005	110000.00	2020-11-05	1.30	6.50
6	81510005	P006	160000.00	2024-07-22	1.30	10.20
7	81510005	P007	120000.00	2022-03-16	1.30	7.00
8	81510006	P008	92000.00	2021-02-28	1.30	5.00
9	81510007	P009	115000.00	2020-08-18	1.30	6.80
10	81510008	P010	90000.00	2024-05-01	1.30	5.50
11	81510009	P011	170000.00	2017-12-01	1.30	11.00
12	81510009	P012	125000.00	2021-04-12	1.30	7.20
13	81510010	P013	118000.00	2024-09-25	1.30	6.90

R3TURNS

	MarketValue	InceptionDate	AnnualizedReturn
1	88000.00	2022-01-10	4.80
2	90000.00	2024-05-01	5.50
3	92000.00	2021-02-28	5.00
4	95000.00	2020-03-15	5.20
5	105000.00	2021-06-20	6.00
6	110000.00	2020-11-05	6.50
7	115000.00	2020-08-18	6.80
8	118000.00	2024-09-25	6.90
9	120000.00	2022-03-16	7.00
10	125000.00	2021-04-12	7.20
11	150000.00	2019-09-01	9.80
12	160000.00	2024-07-22	10.20
13	170000.00	2017-12-01	11.00

STOCK_IN_PORTFOLIO

	ID	PID	Phone	StartDate	AllocationRatio	PostTradeCO	AssetID
1	SIP001	P003	81510002	2019-09-02	70.00	Tiger Brokers	STK101
2	SIP002	P006	81510005	2018-07-23	45.00	Saxo Markets	STK101
3	SIP003	P011	81510009	2017-12-02	75.00	FSMOne	STK202
4	SIP004	P002	81510001	2021-06-21	0.00	Tiger Brokers	STK303
5	SIP005	P007	81510005	2022-03-17	60.00	Saxo Markets	STK010

BOND_IN_PORTFOLIO

	ID	PID	Phone	StartDate	AllocationRatio	PostTradeCO	AssetID
1	BIP001	P001	81510001	2020-03-16	70.00	DBS Bonds	BND170
2	BIP002	P001	81510001	2020-03-16	30.00	OCBC Bonds	BND444
3	BIP003	P004	81510003	2022-01-11	80.00	UOB Bonds	BND001
4	BIP004	P004	81510003	2022-01-11	20.00	Citibank Bonds	BND444
5	BIP005	P005	81510004	2020-11-06	100.00	Tiger Bonds	BND170
6	BIP006	P008	81510006	2021-03-01	65.00	Saxo Bonds	BND001
7	BIP007	P008	81510006	2021-03-01	35.00	FSMOne Bonds	BND444
8	BIP008	P009	81510007	2020-08-19	50.00	Phillip Bonds	BND170
9	BIP009	P010	81510008	2022-05-02	75.00	Maybank Bonds	BND001
10	BIP010	P010	81510008	2022-05-02	25.00	Interactive Bonds	BND444
11	BIP011	P012	81510009	2021-04-13	100.00	DBS Bonds	BND170
12	BIP012	P013	81510010	2020-09-26	100.00	OCBC Bonds	BND170

FUND_IN_PORTFOLIO

	ID	PID	Phone	StartDate	AllocationRatio	PostTradeCO	AssetID
1	FIP001	P002	81510001	2021-06-22	65.00	Tiger Funds	FND111
2	FIP002	P002	81510001	2021-06-22	35.00	Interactive Funds	FND555
3	FIP003	P003	81510002	2019-09-03	30.00	Saxo Funds	FND666
4	FIP004	P005	81510004	2020-11-07	0.00	FSMOne Funds	FND555
5	FIP005	P006	81510005	2018-07-24	55.00	DBS Funds	FND666
6	FIP006	P007	81510005	2022-03-18	40.00	OCBC Funds	FND111
7	FIP007	P009	81510007	2020-08-20	50.00	UOB Funds	FND555
8	FIP008	P011	81510009	2017-12-03	25.00	Citibank Funds	FND666
9	FIP009	P012	81510009	2021-04-14	0.00	Tiger Funds	FND111
10	FIP010	P013	81510010	2020-09-27	0.00	Saxo Funds	FND555

INVESTED_VALUE

	Phone	PID	Date	Amount
1	81510001	P001	2023-01-01	95000.00
2	81510001	P002	2023-01-01	105000.00
3	81510002	P003	2023-01-01	150000.00
4	81510003	P004	2023-01-01	88000.00
5	81510004	P005	2023-01-01	110000.00
6	81510005	P006	2023-01-01	160000.00
7	81510005	P007	2023-01-01	120000.00
8	81510006	P008	2023-01-01	92000.00
9	81510007	P009	2023-01-01	115000.00
10	81510008	P010	2023-01-01	90000.00
11	81510009	P011	2023-01-01	170000.00
12	81510009	P012	2023-01-01	125000.00
13	81510010	P013	2023-01-01	118000.00

UNREALIZED_GAIN_LOSS

	Phone	PID	Date	Amount
1	81510001	P001	2024-01-01	100.00
2	81510001	P001	2024-02-04	100.00
3	81510001	P001	2024-03-05	10.00
4	81510001	P001	2024-03-15	200.00
5	81510001	P001	2024-05-01	100.00
6	81510001	P001	2024-06-02	-80.00
7	81510001	P001	2024-07-02	-100.00
8	81510001	P001	2024-08-03	-1900.00
9	81510001	P001	2024-09-01	1000.00
10	81510001	P001	2024-10-01	-20.00
11	81510001	P001	2024-11-01	-2.00
12	81510001	P001	2024-12-04	10.00
13	81510001	P002	2024-04-01	-20.00
14	81510001	P002	2024-04-10	-20.00
15	81510002	P003	2024-11-01	-15000.00
16	81510003	P004	2024-05-01	1000.00
17	81510004	P005	2024-05-01	4000.00
18	81510005	P006	2024-07-01	-1000.00
19	81510005	P007	2024-07-01	6000.00
20	81510006	P008	2024-08-01	1500.00
21	81510007	P009	2024-03-01	-5000.00
22	81510008	P010	2024-02-01	2500.00
23	81510009	P011	2024-09-01	-20000.00
24	81510009	P012	2024-09-01	7000.00
25	81510010	P013	2024-12-01	6500.00

TRANSACTION1

	ID	Date	PID	Phone	Type
1	T001	2024-03-15	P001	81510001	Buy
2	T002	2024-04-10	P002	81510001	Rebalance
3	T003	2024-05-20	P003	81510002	Sell
4	T004	2024-06-01	P004	81510003	Buy
5	T005	2024-06-15	P005	81510004	Topup
6	T006	2024-07-01	P006	81510005	Sell
7	T007	2024-07-15	P007	81510005	Rebalance
8	T008	2024-08-01	P008	81510006	Buy
9	T009	2024-08-15	P009	81510007	Topup
10	T010	2024-09-01	P010	81510008	Sell
11	T011	2024-09-15	P011	81510009	Buy
12	T012	2024-10-01	P012	81510009	Rebalance
13	T013	2024-10-15	P013	81510010	Topup
14	T014	2024-01-05	P002	81510001	Topup
15	T015	2024-02-04	P002	81510001	Topup
16	T016	2024-03-02	P002	81510001	Topup
17	T017	2024-04-05	P002	81510001	Topup
18	T018	2024-05-01	P002	81510001	Topup
19	T019	2024-06-02	P002	81510001	Topup
20	T020	2024-07-02	P002	81510001	Topup
21	T021	2024-08-03	P002	81510001	Topup
22	T022	2024-09-01	P002	81510001	Topup
23	T023	2024-10-01	P002	81510001	Topup
24	T024	2024-11-01	P002	81510001	Topup
25	T025	2024-12-04	P002	81510001	Topup

TRANSACTION_FEES

	Type	Fee
1	Buy	0.50
2	Rebalance	0.40
3	Sell	0.70
4	Topup	0.20

SQL Queries from Appendix

Query 1

Find investors who are making on average a loss across all their portfolios in 2024.

Assumptions:

- “Loss” is characterised by negative values in the UNREALIZED_GAIN_LOSS.Amount field.
- 0 gain or loss is not a loss.

Explanation:

Steps to compute the required results:

1. SELECT clause: We want to display the phone, name and the average loss of the investors

```
SELECT
  i.Phone,
  i.Name,
  AVG(COALESCE(u.Amount, 0)) AS AvgLoss
```

2. FROM clause: We will join UNREALIZED_GAIN_LOSS with PORTFOLIO1 and join PORTFOLIO1 with INVESTOR

```
FROM
  INVESTOR AS i, UNREALIZED_GAIN_LOSS AS u, PORTFOLIO1 AS p
```

3. WHERE clause: First, we join UNREALIZED_GAIN_LOSS with PORTFOLIO1 using the attributes Phone and PID. Secondly, we join PORTFOLIO1 with INVESTOR using the attribute Phone. Lastly, we choose the records in 2024.

```
WHERE
  i.Phone = p.Phone
  AND p.Phone = u.Phone
  AND p.PID = u.PID
  AND YEAR(u.Date) = 2024
```

4. GROUP BY clause: We group by each investor using the attributes Phone and Name.

```
GROUP BY
  i.Phone, i.Name
```

5. HAVING clause: For each investor, we find those who have an average amount of less than 0. Using COALESCE ensures any if any record is missing, it is treated as if the amount is 0.

```
HAVING
  AVG(COALESCE(u.Amount, 0)) < 0;
```

Final query:

```
SELECT
  i.Phone,
  i.Name,
  AVG(COALESCE(u.Amount, 0)) AS AvgLoss
FROM
  INVESTOR AS i, UNREALIZED_GAIN_LOSS AS u, PORTFOLIO1 AS p
WHERE
  i.Phone = p.Phone
  AND p.Phone = u.Phone
  AND p.PID = u.PID
  AND YEAR(u.Date) = 2024
GROUP BY
  i.Phone, i.Name
HAVING
  AVG(COALESCE(u.Amount, 0)) < 0;
```

Output:

	Phone	Name	AvgLoss
1	81510001	Sherlock Holmes	-44.428571
2	81510002	Harry Potter	-15000.000000
3	81510007	Victor Frankenstein	-5000.000000
4	81510009	John Reacher	-6500.000000

Query 2

Find investors who are seeing an annualized return of more than 10% from their portfolios in 2024.

Assumptions:

- The PORTFOLIO1 table contains the AnnualizedReturn, InceptionDate, PID, and Phone.
- The portfolio is considered active in 2024 if InceptionDate <= '2024-12-31'
- AnnualizedReturn reflects the return percentage relevant to 2024.
- The INVESTOR table is used to retrieve investor names and contact details.

Explanation:

Steps to compute the required results.

1. SELECT specifies the columns to include in the results. p.Phone gets the investor's phone number, i.Name fetches the investor's name from the INVESTOR table, p.PID returns the portfolio ID and p.AnnualizedReturn displays the return percentage.

```
SELECT
  p.Phone, -- Investor's phone number
  i.Name, -- Investor's name
  p.PID, -- Portfolio ID
  p.AnnualizedReturn -- Annualized return %
```

2. FROM clause identifies the primary table (PORTFOLIO1) and uses a JOIN with INVESTOR to get investor information. The WHERE clause filters to include only portfolios with AnnualizedReturn > 10% and only portfolios that were active in 2024.

```
FROM
  PORTFOLIO1 p
JOIN
  INVESTOR i ON p.Phone = i.Phone
WHERE
  p.AnnualizedReturn > 10 -- Return greater than 10%
  AND p.InceptionDate <= '2024-12-31' -- Portfolio active in 2024
```

3. The ORDER BY clause is optional but can be included for sorted results.

```
ORDER BY
  p.AnnualizedReturn DESC;
```

Final query:

```
SELECT
  p.Phone, -- Investor's phone number
  i.Name, -- Investor's name
  p.PID, -- Portfolio ID
  p.AnnualizedReturn -- Annualized return %
FROM
  PORTFOLIO1 p
JOIN
  INVESTOR i ON p.Phone = i.Phone
WHERE
  p.AnnualizedReturn > 10 -- Return greater than 10%
  AND p.InceptionDate <= '2024-12-31' -- Portfolio active in 2024
ORDER BY
  p.AnnualizedReturn DESC; -- Highest return first
```

Output:

	Phone	Name	PID	AnnualizedReturn
1	81510009	John Reacher	P011	11.00
2	81510005	Jay Gatsby	P006	10.20

Query 3

Find the monthly average unrealized gain/loss of portfolios for each month in 2024.

Assumptions:

- The UNREALIZED_GAIN_LOSS table contains transactions exclusively for 2024.
- The Amount field uses negative values to denote losses and positive values for gains.
- The Date column is stored in a valid date format, allowing date-related functions (FORMAT, MONTH, DATENAME) to work correctly

Explanation:

Steps to compute the required results.

1. SELECT specifies the columns to include in the results. It extracts the year and month from the Date column using FORMAT(Date, 'yyyy-MM') and aliases it as YearMonth, the month number with MONTH(Date) as MonthNumber, and the full month name with DATENAME(MONTH, Date) as MonthName. Additionally, AVG(Amount) calculates the average of the Amount column for each group, aliased as AverageUnrealizedGainLoss.

```
SELECT
  FORMAT(Date, 'yyyy-MM') AS YearMonth, -- extract month and year as YearMonth
  MONTH(Date) AS MonthNumber, -- month clause gets the month number from Date
  DATENAME(MONTH, Date) AS MonthName, -- DATENAME gets the monthname
  AVG(Amount) AS AverageUnrealizedGainLoss -- average of gain/loss
),
```

2. The FROM clause specifies the source table, which is UNREALIZED_GAIN_LOSS, where the data is being queried from. The WHERE clause filters records to include only unrealized gain/loss transactions from the year 2024.

```
FROM
  UNREALIZED_GAIN_LOSS
WHERE
  YEAR(Date) = 2024 -- in 2024
```

3. The GROUP BY clause results by YearMonth (formatted as yyyy-MM), MonthNumber, and MonthName to aggregate data at the monthly level and compute the average gain/loss for each month. Results are ordered by the Month Number (ascending).

```
GROUP BY
  FORMAT(Date, 'yyyy-MM'), -- Grouped by
  Month
  MONTH(Date),
  DATENAME(MONTH, Date)
ORDER BY
  MonthNumber; -- Runs from Jan to Dec
```

Final Query:

```
SELECT
  FORMAT(Date, 'yyyy-MM') AS YearMonth, -- extract month and year as YearMonth
  MONTH(Date) AS MonthNumber, -- month clause gets the month number from Date
  DATENAME(MONTH, Date) AS MonthName, -- DATENAME gets the monthname
  AVG(Amount) AS AverageUnrealizedGainLoss -- average of gain/loss
FROM
  UNREALIZED_GAIN_LOSS
WHERE
  YEAR(Date) = 2024 -- in 2024
GROUP BY
  FORMAT(Date, 'yyyy-MM'), -- Grouped by Month
  MONTH(Date),
  DATENAME(MONTH, Date)
ORDER BY
  MonthNumber; -- Runs from Jan to Dec
```

Output:

	YearMonth	MonthNumber	MonthName	AverageUnrealizedGainLoss
1	2024-01	1	January	100.000000
2	2024-02	2	February	1300.000000
3	2024-03	3	March	-1596.666666
4	2024-04	4	April	-20.000000
5	2024-05	5	May	1700.000000
6	2024-06	6	June	-80.000000
7	2024-07	7	July	1633.333333
8	2024-08	8	August	-200.000000
9	2024-09	9	September	-4000.000000
10	2024-10	10	October	-20.000000
11	2024-11	11	November	-7501.000000
12	2024-12	12	December	3255.000000

Query 4

What are the top three most popular first financial goals for investors in 2024?

Assumptions:

- Identify investors whose first portfolio started in 2024.
- Exclude any investors who have portfolios prior to 2024.
- All the goals still in the table are assumed to have a portfolio currently running. Any portfolio that has expired or been closed, the associated goal will be removed.

Explanation:

Steps to compute the required result:

1. Create a temporary view to store InceptionDate of each Investor's Portfolio which are currently open (2024).

```
WITH FIRST_PORTFOLIO AS (  
  SELECT Phone, MIN(InceptionDate) AS FirstPortfolioDate  
  FROM PORTFOLIO1  
  GROUP BY Phone  
  -- ING MIN(InceptionDate) BETWEEN '2024-01-01' AND '2024-12-31'  
)
```

2. SELECT the financial goals and the count so top entries can be output as the result.

```
SELECT TOP 3 fg.Goal, COUNT(*) AS InvestorCount
```

3. Selecting FROM the Goal from the FINANCIAL_GOAL table and joining it with the previously created temporary view and investor.

```
FROM FINANCIAL_GOAL AS fg, FIRST_PORTFOLIO AS fp, INVESTOR AS i
```

4. WHERE constraints are used to join those entries of FINANCIAL_GOAL which have a valid Investor and Portfolio(s).

```
WHERE fg.Phone = fp.Phone AND i.Phone = fg.Phone
```

5. GROUP BY Goal to ensure only unique entries are listed and ORDER BY descending order.

```
GROUP BY fg.Goal
```



```
ORDER BY InvestorCount DESC;
```

Final Query:

```
WITH FIRST_PORTFOLIO AS (  
  SELECT Phone, MIN(InceptionDate) AS FirstPortfolioDate  
  FROM PORTFOLIO1  
  GROUP BY Phone  
  -- ING MIN(InceptionDate) BETWEEN '2024-01-01' AND '2024-12-31'  
)  
SELECT TOP 3 fg.Goal, COUNT(*) AS InvestorCount  
FROM FINANCIAL_GOAL AS fg, FIRST_PORTFOLIO AS fp, INVESTOR AS i  
WHERE fg.Phone = fp.Phone AND i.Phone = fg.Phone  
GROUP BY fg.Goal  
ORDER BY InvestorCount DESC;
```

Output:

	Goal	NumberOfInvestors
1	Retirement	4
2	Child Education	3
3	House Purchase	2

Query 5

Find investors who consistently top up their investment at the beginning of every month (dollar-cost averaging) in 2024 for at least one of their portfolios.

Assumptions:

- The Date attribute in TRANSACTION1 is saved as DD.MM.YYYY.
- Consistent top-up in this scenario is where all 12 months of 2024 have at least one top-up in the portfolio.
- Identify investors by their name and phone number.
- Assuming "beginning of month" means within first 5 days.

Explanation:

This query uses 3 CTEs to identify the portfolios with consistent top-ups, then determine the investors of those portfolios.

Steps to compute the required result:

1. Identify all portfolios with top-up transactions in 2024

```
WITH MonthlyTopups AS (  
  SELECT  
    t.Phone,  
    t.PID,  
    MONTH(t.Date) AS MonthNumber,  
    COUNT(*) AS TopupCount  
  FROM  
    Transaction1 t  
  WHERE  
    t.Type = 'TOPUP'  
    AND YEAR(t.Date) = 2024  
    AND DAY(t.Date) <= 5  
  GROUP BY  
    t.Phone,  
    t.PID,  
    MONTH(t.Date)  
)
```

Using WITH, create a temporary view called Monthly_TopUps from TRANSACTION1 and identify top-up transactions done in 2024.

For the WHERE Condition, We use t.type=top_up, as we previously defined TRANSACTION1.type to be top_up, withdrawal, etc.

We check for any top-ups for the first 5 days of the month, and group by the portfolio, investor, and the month

MONTH() returns a number 1-12, for which month the top-up happens in.

2. Identify all portfolios with Consistent top-up transactions in 2024

```
SELECT  
  i.Name AS InvestorName,  
  i.Phone,  
  COUNT(DISTINCT m.MonthNumber) AS  
  MonthsWithTopups  
FROM  
  INVESTOR i  
JOIN  
  MonthlyTopups m ON i.Phone = m.Phone  
GROUP BY  
  i.Name,
```

```
i.Phone  
HAVING  
COUNT(DISTINCT m.MonthNumber) = 12  
ORDER BY  
MonthsWithTopups DESC;
```

SELECT the Investors' name, phone and number of monthly top ups in 2024.
Since we only want consistent top-ups for the whole year, we set COUNT = 12.
GROUP BY helps to group up all the portfolios and outputs investors that have at least 1 portfolio with monthly top-ups=12.

Final Query:

```
WITH MonthlyTopups AS (  
  SELECT  
    t.Phone,  
    t.PID,  
    MONTH(t.Date) AS MonthNumber,  
    COUNT(*) AS TopupCount  
  FROM  
    Transaction1 t  
  WHERE  
    t.Type = 'TOPUP'  
    AND YEAR(t.Date) = 2024  
    AND DAY(t.Date) <= 5 -- Assuming "beginning of month" means within first 5 days  
  GROUP BY  
    t.Phone,  
    t.PID,  
    MONTH(t.Date)  
)  
SELECT  
  i.Name AS InvestorName,  
  i.Phone,  
  COUNT(DISTINCT m.MonthNumber) AS MonthsWithTopups  
FROM  
  INVESTOR i  
JOIN  
  MonthlyTopups m ON i.Phone = m.Phone  
GROUP BY
```

```

i.Name,
i.Phone
HAVING
COUNT(DISTINCT m.MonthNumber) =12
ORDER BY
MonthsWithTopups DESC;

```

Output:

	InvestorName	Phone	MonthsWithTopups
1	Sherlock Holmes	81510001	12

Query 6

Find the most popular financial goals for investors working in the same company and whose age is between 30 to 40 years old.

Assumptions:

- Age is calculated based on the difference between current year and investor's date of birth.
- Only investors aged between 30 and 40 (inclusive) are considered.
- Each investor may have one or more financial goals in the FINANCIAL_GOAL table.
- We assume the goal is "popular" if it has the highest count per company — although
- This query shows all goals grouped by company, not just the top one per company.

Explanation:

This query uses two CTEs to filter eligible investors by age and then aggregates their financial goals by company.

Steps to compute the required results.

1. Compute Investor Age

```

WITH InvestorAge AS (
SELECT
i.Phone,
i.Name,
i.Company,
i.DoB,
DATEDIFF(YEAR, i.DoB, GETDATE()) AS Age
FROM
INVESTOR i
),

```

InvestorAge calculates each investor's age by subtracting their date of birth from the current date (GETDATE()), using DATEDIFF(YEAR, ...). It keeps track of their name, phone, and company for future use.

2. Filter Investors based on Age

```
AgeFiltered AS (  
  SELECT  
    ia.Phone,  
    ia.Name,  
    ia.Company,  
    ia.Age  
  FROM  
    InvestorAge ia  
  WHERE  
    ia.Age BETWEEN 30 AND 40  
)
```

AgeFiltered filters out only those investors whose computed age falls between 30 and 40 years old.

3. Aggregate Financial Goals by Company

```
SELECT  
  af.Company,  
  fg.Goal,  
  COUNT(*) AS NumberOfInvestors  
FROM  
  AgeFiltered af  
JOIN  
  FINANCIAL_GOAL fg ON af.Phone = fg.Phone  
GROUP BY  
  af.Company,  
  fg.Goal  
ORDER BY  
  af.Company,  
  NumberOfInvestors DESC;
```

Joins the filtered investors with their financial goals using Phone as the key, groups by company and goal to count how many investors at each company share the same goal and orders the results by company and popularity of the goal (descending count).

Final Query:

```
WITH InvestorAge AS (  
  SELECT
```

```

i.Phone,
i.Name,
i.Company,
i.DoB,
DATEDIFF(YEAR, i.DoB, GETDATE()) AS Age
FROM
INVESTOR i
),
AgeFiltered AS (
SELECT
ia.Phone,
ia.Name,
ia.Company,
ia.Age
FROM
InvestorAge ia
WHERE
ia.Age BETWEEN 30 AND 40
)
SELECT
af.Company,
fg.Goal,
COUNT(*) AS NumberOfInvestors
FROM
AgeFiltered af
JOIN
FINANCIAL_GOAL fg ON af.Phone = fg.Phone
GROUP BY
af.Company,
fg.Goal
ORDER BY
af.Company,
NumberOfInvestors DESC;

```

Output:

	Company	Goal	NumberOfInvestors
1	DBS	Retirement	2
2	DBS	House Purchase	1
3	DBS	Property Purchase	1
4	OCBC	Child Education	2
5	OCBC	House Purchase	1
6	OCBC	Lambo Purchase	1
7	OCBC	Retirement	1
8	OCBC	Rolex Collection	1

Query 7

Are male investors in their 20s making more money from their investments than their female counterparts in 2024?

Assumptions:

- Making more money is measured by having higher average unrealised gain across portfolios in 2024.
- We only consider the investor's age as of 1/1/2024.
- Age is calculated using the DATEDIFF(YEAR, i.DoB, '2024-01-01') function, this includes the investors aged 20-29.

Explanation:

Steps to compute the required results:

1. SELECT clause: We want to display the 2 genders and its corresponding average gain in the output.

```
SELECT
  i.Gender,
  AVG(COALESCE(u.Amount, 0)) AS AvgGain
```

2. FROM clause: We will join UNREALIZED_GAIN_LOSS with PORTFOLIO1 and join PORTFOLIO1 with INVESTOR.

```
FROM
  INVESTOR i, UNREALIZED_GAIN_LOSS AS u, PORTFOLIO1 AS p
```

3. WHERE clause: First, we join INVESTOR and PORTFOLIO1 using the attribute Phone. Secondly, we join PORTFOLIO1 and UNREALIZED_GAIN_LOSS using the attributes Phone and PID. Thirdly, we only consider records from 2024. Lastly, we only want records where the age of investors are 20-29.

```
WHERE
  i.Phone = p.Phone
  AND p.Phone = u.Phone
  AND p.PID = u.PID
  AND YEAR(u.Date) = 2024
  AND TIMESTAMPDIFF(YEAR, i.DoB, '2024-01-01') BETWEEN 20 AND 29
```

4. GROUP BY clause: This groups the results by gender so the query calculates the average gain for each gender.

```
GROUP BY
  i.Gender;
```

Final query:

```
SELECT
  i.Gender,
  AVG(COALESCE(u.Amount, 0)) AS AvgGainLoss
FROM
  INVESTOR i, UNREALIZED_GAIN_LOSS AS u, PORTFOLIO1 AS p
WHERE
  i.Phone = p.Phone
  AND p.Phone = u.Phone
  AND p.PID = u.PID
  AND YEAR(u.Date) = 2024
  AND DATEDIFF(YEAR, i.DoB, '2024-01-01') BETWEEN 20 AND 29
GROUP BY
  i.Gender;
```

Output:

	Gender	AvgGainLoss
1	F	-3500.000000
2	M	6500.000000

Additional Query 1

Which Investors have more than 2 portfolios?

Assumptions:

- The PORTFOLIO1 table contains the contact details.
- The INVESTOR table is used to retrieve investor names and contact details.

Explanation:

Steps to compute the required results:

1. SELECT clause displays the investor's name and phone and their corresponding number of portfolios.

```
SELECT  
i.Name, i.Phone, COUNT(PID) AS NumPortfolios
```

2. FROM clause identifies the primary table (PORTFOLIO1) and uses a JOIN with INVESTOR to get investor information.

```
FROM  
PORTFOLIO1 p  
JOIN  
INVESTOR i ON p.Phone = i.Phone
```

3. GROUP BY clause groups the results by the investor's phone number and investor's name.

```
GROUP BY  
i.Phone, i.Name
```

4. HAVING clause counts the corresponding number of portfolios using PID for each investor, and returns the investors with more than 2 portfolios.

```
HAVING COUNT(PID) > 2
```

5. ORDER BY clause is optional but can be added for sorted results.

```
ORDER BY NumPortfolios DESC;
```

Final query:

```
SELECT
```

```

i.Name, i.Phone, COUNT(PID) AS NumPortfolios
FROM
  PORTFOLIO1 p
JOIN
  INVESTOR i ON p.Phone = i.Phone
GROUP BY
  i.Phone, i.Name
HAVING COUNT(PID) > 2
ORDER BY NumPortfolios DESC;

```

Additional Query 2

Which Investors have made an unrealised gain of 20% of their total investment amount so far?

Assumptions:

- Total investment is equal to the sum of total invested value of all portfolios for an Investor.
- Total unrealised gain is equal to the sum of the net unrealised gain/loss of all portfolios for an investor.

Explanation:

Steps to compute the required results:

1. Create a temporary view to aggregate the investors' total investment value across all their portfolios and determine 20% of that sum.

```

WITH 20Percent_Total_Investment AS (
  SELECT Phone, SUM(Amount) * 0.20 AS Gains
  FROM INVESTED_VALUE iv
  GROUP BY Phone
)

```

2. SELECT Name and unique Phone number for each Investor.

```

SELECT i.Name, i.Phone

```

3. Join UNREALIZED_GAIN_LOSS, INVESTOR and the newly created temporary view 20Percent_Total_Investment. Join the tables on the Phone attribute. Only compare the Phone attribute for each UNREALIZED_GAIN_LOSS and 20Percent_Total_Investment to ensure the unrealized gain/loss and invested value of each PORTFOLIO1 belonging to the INVESTOR is summed.

```
FROM UNREALIZED_GAIN_LOSS AS ugl, INVESTOR AS i, 20Percent_Total_Investment AS pti
WHERE i.Phone = ugl.Phone AND ugl.Phone = pti.Phone
```

4. GROUP BY the i.Name, i.Phone, pti.Gains ensure that the unrealized gain/loss matches 20% of the total Invested Amount.

```
GROUP BY i.Name, i.Phone, pti.Gains
HAVING SUM(uglify.Amount) = pti.Gains
```

Final query:

```
WITH Percent_Total_Investment AS (
  SELECT Phone, SUM(Amount) * 0.20 AS Gains
  FROM INVESTED_VALUE iv
  GROUP BY Phone
)
SELECT i.Name, i.Phone
FROM UNREALIZED_GAIN_LOSS AS ugl, INVESTOR AS i, Percent_Total_Investment AS pti
WHERE i.Phone = ugl.Phone AND ugl.Phone = pti.Phone
GROUP BY i.Name, i.Phone, pti.Gains
HAVING SUM(uglify.Amount) = pti.Gains
```