Mutual Fund Tracking System

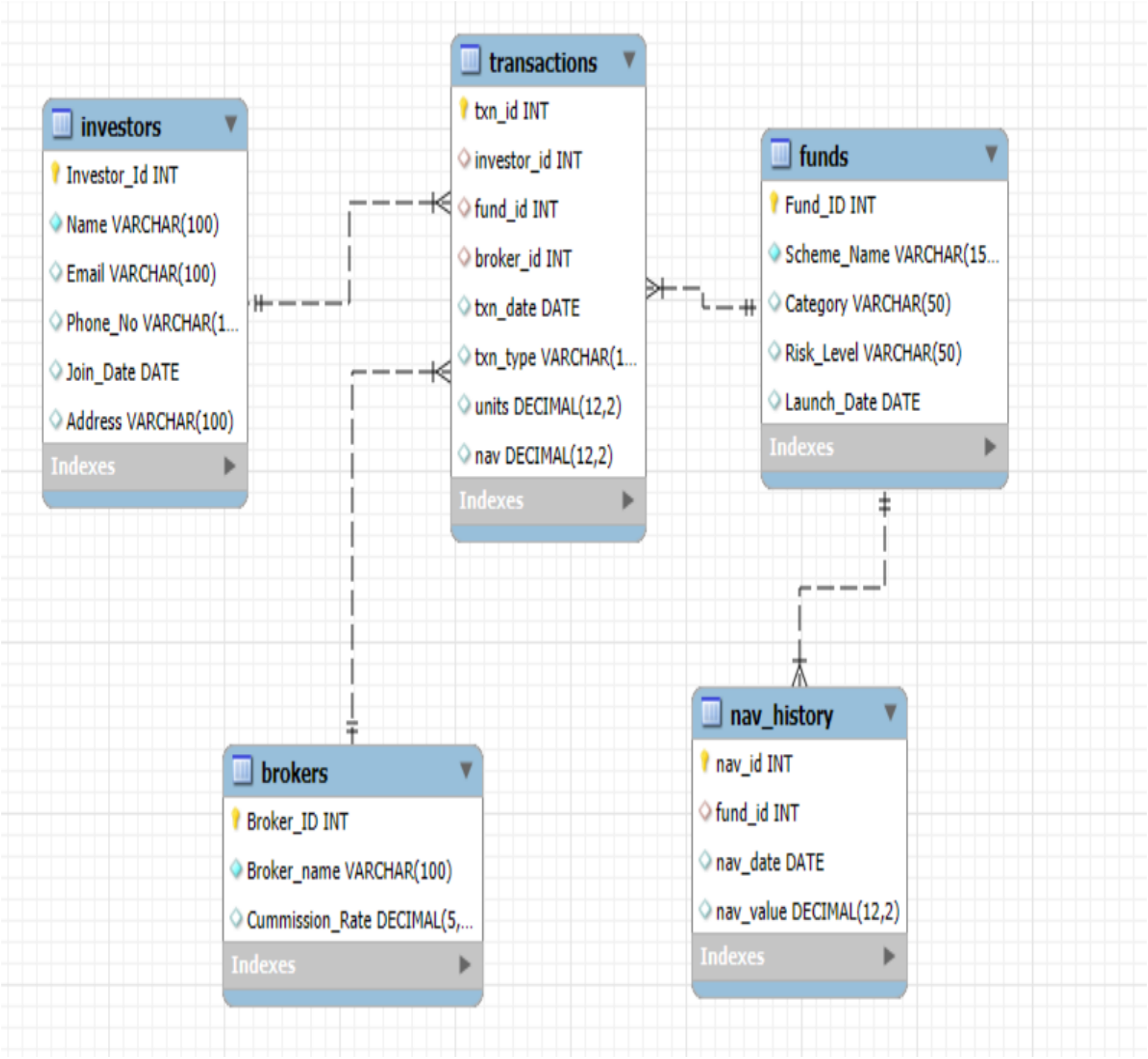
INTRODUCTION

The **Mutual Fund Investment Tracking System** is designed to help manage and analyze investors mutual fund portfolios efficiently. It maintains detailed information about investors, funds, brokers, transactions, and NAV (Net Asset Value) history. The system enables investors and administrators to track investments, monitor portfolio performance, and generate insights for better decision-making.

From a technical perspective, this SQL-based project demonstrates the use of **relational database concepts** such as creating tables, defining relationships, and enforcing constraints. It also includes **SQL operations** like Data Definition Language (DDL), Data Manipulation Language (DML), and Data Query Language (DQL) commands, along with advanced concepts such **as joins, subqueries, aggregate functions, and views.**

The project provides practical exposure to real-world database handling by simulating how mutual fund transactions are recorded, how NAV values fluctuate over time, and how reports can be generated to analyze investor holdings. This system not only strengthens database management skills but also highlights the importance of structured financial data in investment analysis.

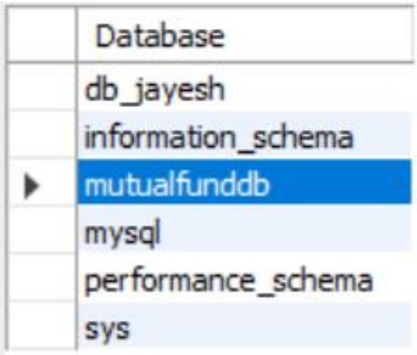
ER Daigram


transactions
txn_id INT
investor_id INT
fund_id INT
broker_id INT
txn_date DATE
txn_type VARCHAR(1...
units DECIMAL(12,2)
nav DECIMAL(12,2)
Indexes
investors
Investor_Id INT
Name VARCHAR(100)
Email VARCHAR(100)
Phone_No VARCHAR(1...
Join_Date DATE
Address VARCHAR(100)
Indexes
funds
Fund_ID INT
Scheme_Name VARCHAR(15...
Category VARCHAR(50)
Risk_Level VARCHAR(50)
Launch_Date DATE
Indexes
nav_history
nav_id INT
fund_id INT
nav_date DATE
nav_value DECIMAL(12,2)
Indexes
brokers
Broker_ID INT
Broker_name VARCHAR(100)
Cummission_Rate DECIMAL(5,...
Indexes

Database: -

CREATE DATABASE MutualFundDB**;**

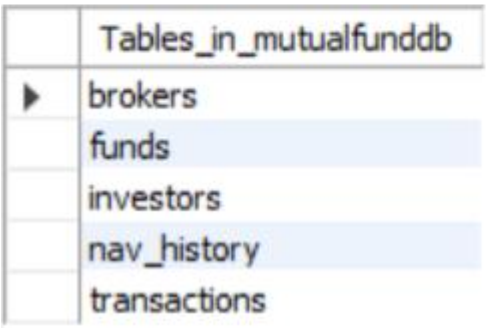
SHOW DATABASES;



Database
db_jayesh
information_schema
mutualfunddb
mysql
performance_schema
sys

Table in MutualFundDB Database: -

SHOW TABLES;



Tables_in_mutualfunddb
brokers
funds
investors
nav_history
transactions

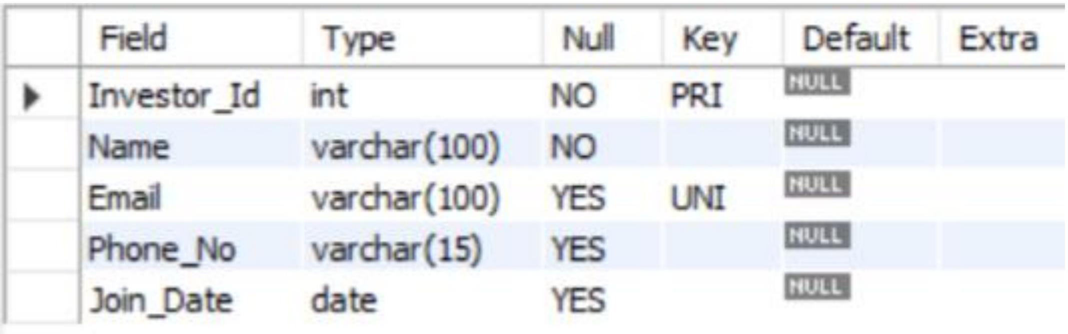
1) DATA DEFINATION LANGUAGE (DDL):

1. Creating Tables: -

A) Investors

CREATE TABLE Investors(Investor_Id INT PRIMARY KEY, Name VARCHAR(100) NOT NULL,Email VARCHAR(100) UNIQUE, Phone_No VARCHAR(15),Join_Date DATE);

DESC Investors;

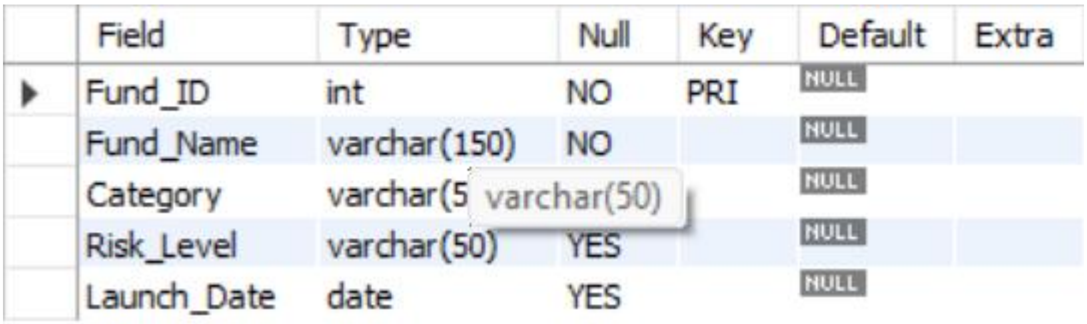


	Field	Type	Null	Key	Default	Extra
▶	Investor_Id	int	NO	PRI	NULL	
	Name	varchar(100)	NO		NULL	
	Email	varchar(100)	YES	UNI	NULL	
	Phone_No	varchar(15)	YES		NULL	
	Join_Date	date	YES		NULL	

B) Funds

CREATE TABLE Funds(Fund_ID INT PRIMARY KEY, Fund_Name VARCHAR(150) NOT NULL,Category VARCHAR(50),Risk_Level VARCHAR(50),Launch_Date DATE);

DESC Funds;

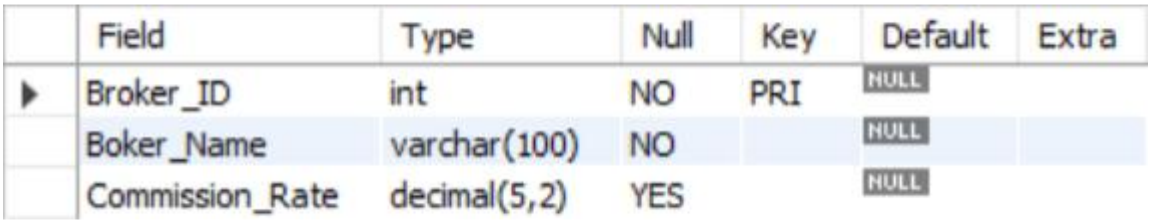


	Field	Type	Null	Key	Default	Extra
▶	Fund_ID	int	NO	PRI	NULL	
	Fund_Name	varchar(150)	NO		NULL	
	Category	varchar(5 varchar(50)			NULL	
	Risk_Level	varchar(50)	YES		NULL	
	Launch_Date	date	YES		NULL	

C) Brokers

```
CREATE TABLE Brokers(Broker_ID INT PRIMARY KEY,Boker_Name
VARCHAR(100) NOT NULL,Commission_Rate DECIMAL(5,2));
```

```
DESC Brokers;
```



	Field	Type	Null	Key	Default	Extra
▶	Broker_ID	int	NO	PRI	NULL	
	Boker_Name	varchar(100)	NO		NULL	
	Commission_Rate	decimal(5,2)	YES		NULL	

D) Transactions

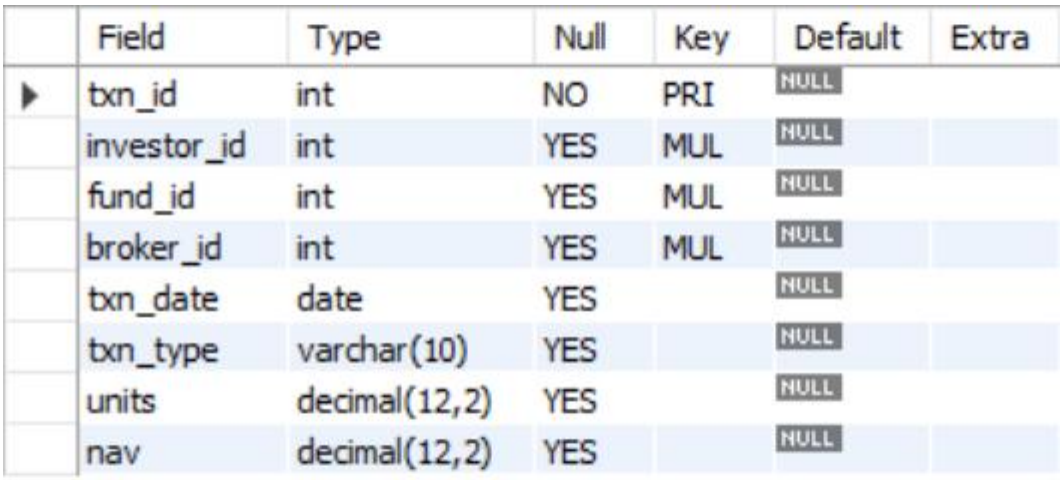
```
CREATE TABLE Transactions (txn_id INT PRIMARY KEY,investor_id
INT,fund_id INT,broker_id INT,txn_date DATE,txn_type VARCHAR(10)
CHECK (txn_type IN ('BUY','SELL')),units DECIMAL(12,2),nav
DECIMAL(12,2),

FOREIGN KEY (investor_id) REFERENCES Investors(investor_id),

FOREIGN KEY (fund_id) REFERENCES Funds(fund_id),

FOREIGN KEY (broker_id) REFERENCES Brokers(broker_id));
```

```
DESC Transations;
```



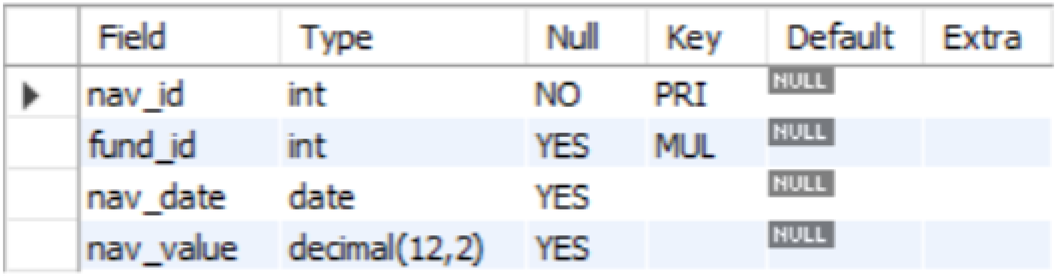
	Field	Type	Null	Key	Default	Extra
▶	txn_id	int	NO	PRI	NULL	
	investor_id	int	YES	MUL	NULL	
	fund_id	int	YES	MUL	NULL	
	broker_id	int	YES	MUL	NULL	
	txn_date	date	YES		NULL	
	txn_type	varchar(10)	YES		NULL	
	units	decimal(12,2)	YES		NULL	
	nav	decimal(12,2)	YES		NULL	

E) NAV_History

```
CREATE TABLE NAV_History (nav_id INT PRIMARY KEY,fund_id
INT,nav_date DATE,nav_value DECIMAL(12,2),

FOREIGN KEY (fund_id) REFERENCES Funds(fund_id));

DESC NAV_History;
```

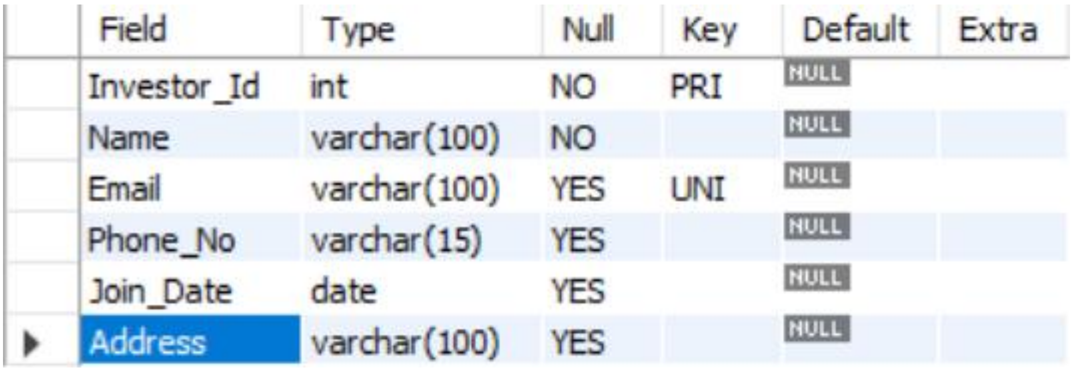


	Field	Type	Null	Key	Default	Extra
▶	nav_id	int	NO	PRI	NULL	
	fund_id	int	YES	MUL	NULL	
	nav_date	date	YES		NULL	
	nav_value	decimal(12,2)	YES		NULL	

2. Alter Table: -

a) Add Column

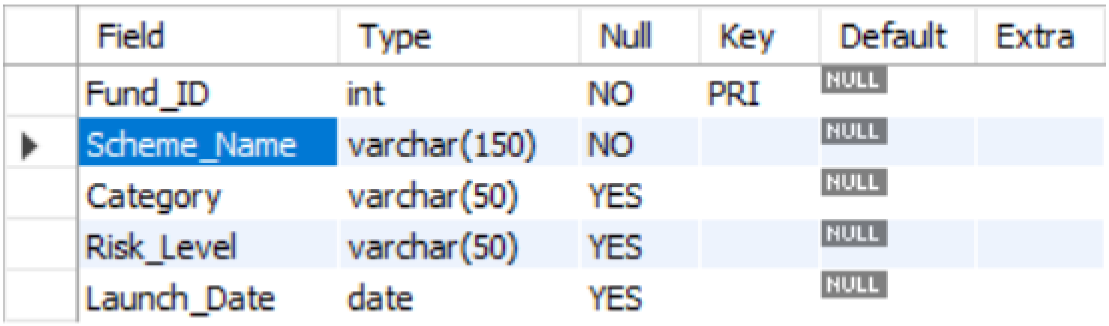
```
ALTER TABLE Investors ADD COLUMN Address VARCHAR(100);
```



	Field	Type	Null	Key	Default	Extra
	Investor_Id	int	NO	PRI	NULL	
	Name	varchar(100)	NO		NULL	
	Email	varchar(100)	YES	UNI	NULL	
	Phone_No	varchar(15)	YES		NULL	
	Join_Date	date	YES		NULL	
▶	Address	varchar(100)	YES		NULL	

B) Rename Column

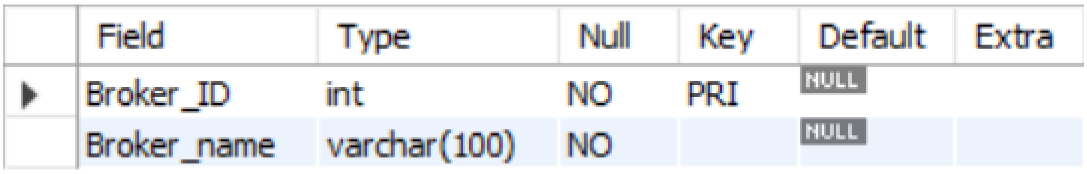
```
ALTER TABLE Funds RENAME COLUMN Fund_Name TO
Scheme_Name;
```



	Field	Type	Null	Key	Default	Extra
	Fund_ID	int	NO	PRI	NULL	
▶	Scheme_Name	varchar(150)	NO		NULL	
	Category	varchar(50)	YES		NULL	
	Risk_Level	varchar(50)	YES		NULL	
	Launch_Date	date	YES		NULL	

C) Drop Column

```
ALTER TABLE Brokers DROP COLUMN Commission_Rate;
```



	Field	Type	Null	Key	Default	Extra
▶	Broker_ID	int	NO	PRI	NULL	
	Broker_name	varchar(100)	NO		NULL	

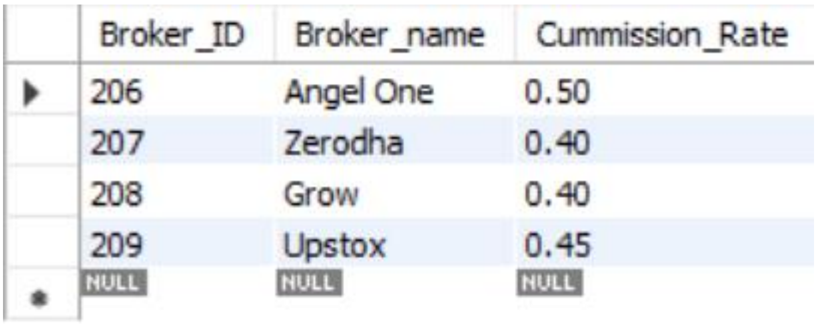
D) Drop Table

```
DROP TABLE NAV_History;
```

2) DATA MANIPULATION LANGUAGE (DML):

1. Insert Into Table: -

INSERT INTO Brokers(broker_id,Broker_Name,Cummission_rate) VALUES (206,'Angel One',0.50);

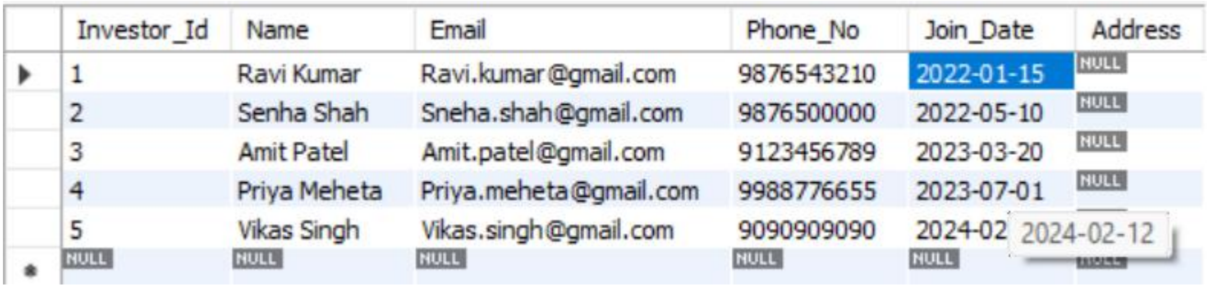


Broker_ID	Broker_name	Cummission_Rate
206	Angel One	0.50
207	Zerodha	0.40
208	Grow	0.40
209	Upstox	0.45
NULL	NULL	NULL

2. Update Into Table: -

Q. Update Join Date Having Investor_ID=1.

UPDATE investors SET join_date = '2022-01-15' WHERE Investor_id = 1;

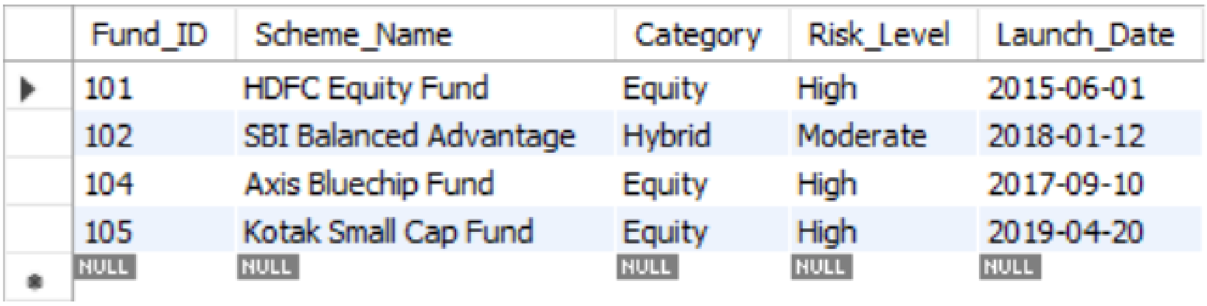


Investor_Id	Name	Email	Phone_No	Join_Date	Address
1	Ravi Kumar	Ravi.kumar@gmail.com	9876543210	2022-01-15	NULL
2	Senha Shah	Sneha.shah@gmail.com	9876500000	2022-05-10	NULL
3	Amit Patel	Amit.patel@gmail.com	9123456789	2023-03-20	NULL
4	Priya Meheta	Priya.meheta@gmail.com	9988776655	2023-07-01	NULL
5	Vikas Singh	Vikas.singh@gmail.com	9090909090	2024-02 2024-02-12	
NULL	NULL	NULL	NULL	NULL	NULL

3. Delete From Table: -

Q. Delete Record from Funds Table Having Fund_ID=103.

DELETE FROM Funds WHERE Fund_ID=103;



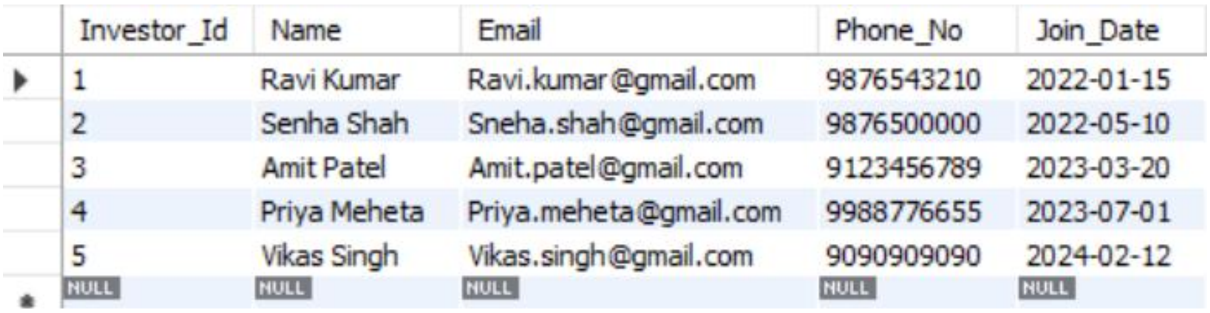
Fund_ID	Scheme_Name	Category	Risk_Level	Launch_Date
101	HDFC Equity Fund	Equity	High	2015-06-01
102	SBI Balanced Advantage	Hybrid	Moderate	2018-01-12
104	Axis Bluechip Fund	Equity	High	2017-09-10
105	Kotak Small Cap Fund	Equity	High	2019-04-20
NULL	NULL	NULL	NULL	NULL

3) DATA QUERY LANGUAGE (DQL):

1) Select Query

a. Selecte Query From entire data.

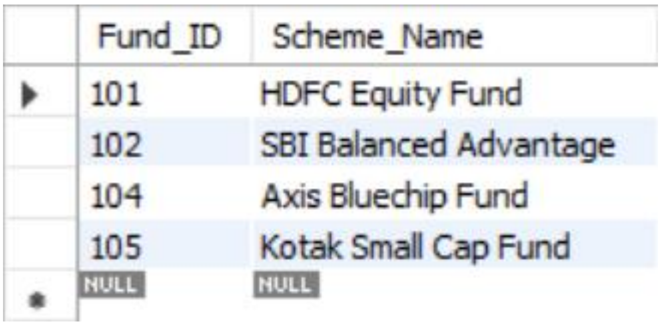
```
SELECT * FROM Investors;
```



Investor_Id	Name	Email	Phone_No	Join_Date
1	Ravi Kumar	Ravi.kumar@gmail.com	9876543210	2022-01-15
2	Senha Shah	Sneha.shah@gmail.com	9876500000	2022-05-10
3	Amit Patel	Amit.patel@gmail.com	9123456789	2023-03-20
4	Priya Meheta	Priya.meheta@gmail.com	9988776655	2023-07-01
5	Vikas Singh	Vikas.singh@gmail.com	9090909090	2024-02-12
NULL	NULL	NULL	NULL	NULL

b. Selecte Specific Data From Fund table.

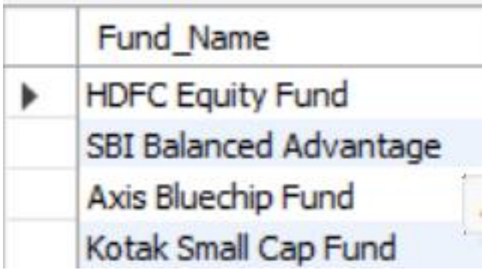
```
SELECT Fund_ID,Scheme_Name FROM Funds;
```



Fund_ID	Scheme_Name
101	HDFC Equity Fund
102	SBI Balanced Advantage
104	Axis Bluechip Fund
105	Kotak Small Cap Fund
NULL	NULL

c. Select Query with changing Column name.

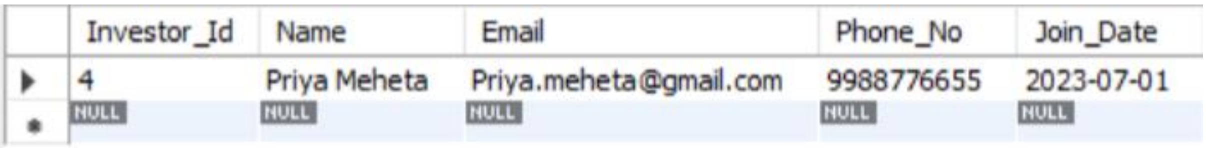
```
SELECT Scheme_Name AS Fund_Name FROM Funds;
```



Fund_Name
HDFC Equity Fund
SBI Balanced Advantage
Axis Bluechip Fund
Kotak Small Cap Fund

d) Show the data having Investor_ID=4.

```
SELECT * FROM Investors WHERE Investor_ID=4;
```

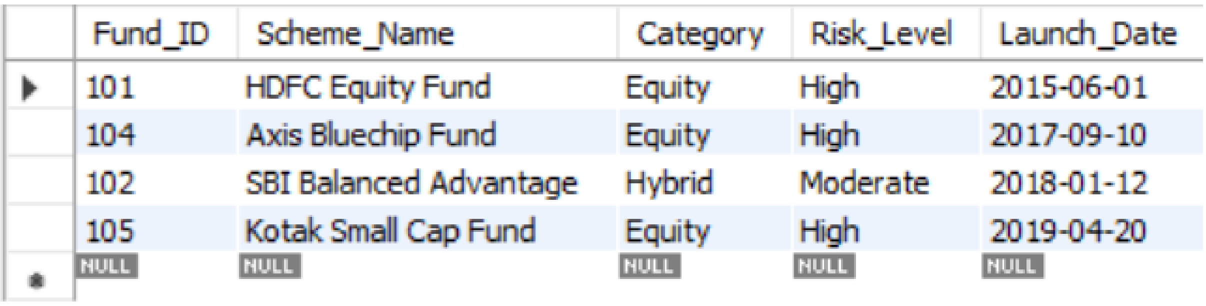


Investor_Id	Name	Email	Phone_No	Join_Date
4	Priya Meheta	Priya.meheta@gmail.com	9988776655	2023-07-01
NULL	NULL	NULL	NULL	NULL

2) Order by

a) List Funds in ascending order by Launch_date.

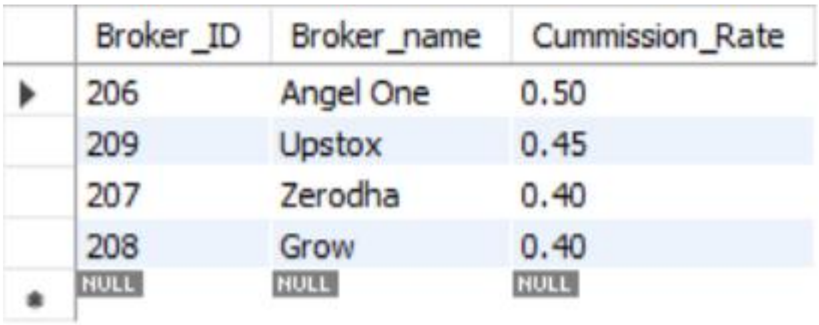
```
SELECT * FROM Funds ORDER BY Launch_date ;
```



Fund_ID	Scheme_Name	Category	Risk_Level	Launch_Date
101	HDFC Equity Fund	Equity	High	2015-06-01
104	Axis Bluechip Fund	Equity	High	2017-09-10
102	SBI Balanced Advantage	Hybrid	Moderate	2018-01-12
105	Kotak Small Cap Fund	Equity	High	2019-04-20
NULL	NULL	NULL	NULL	NULL

b) List Brokres in descending Order by Cummission_rate.

```
SELECT * FROM Brokers ORDER BY Cummission_Rate DESC;
```

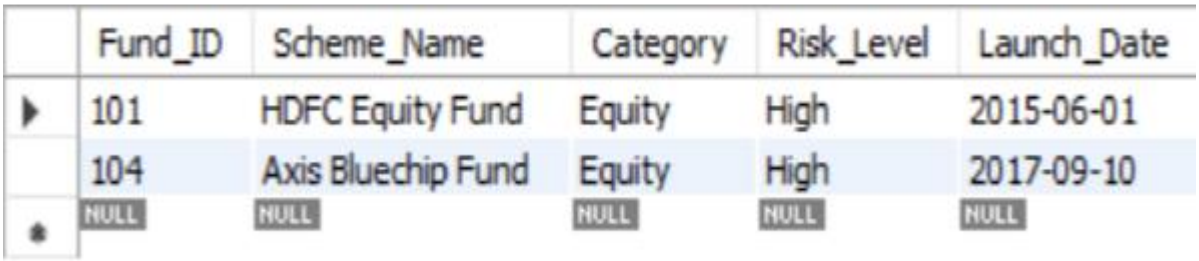


Broker_ID	Broker_name	Cummission_Rate
206	Angel One	0.50
209	Upstox	0.45
207	Zerodha	0.40
208	Grow	0.40
NULL	NULL	NULL

3) Limit Query

Display Top 2 Funds by Launch_date.

SELECT * FROM Funds ORDER BY Launch_Date LIMIT 2;

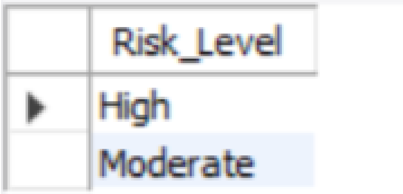


	Fund_ID	Scheme_Name	Category	Risk_Level	Launch_Date
▶	101	HDFC Equity Fund	Equity	High	2015-06-01
	104	Axis Bluechip Fund	Equity	High	2017-09-10
*	NULL	NULL	NULL	NULL	NULL

4) Distinct Query

Display Unique Risk_level From Funds.

SELECT DISTINCT Risk_Level FROM Funds;



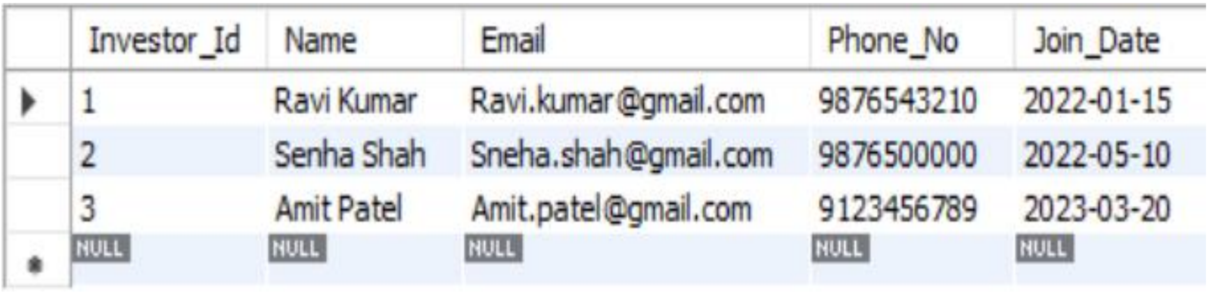
	Risk_Level
▶	High
	Moderate

5) Where Clause

1. With Comparison Operator

Find investor name whose Join date is before ‘2023-07-01’.

SELECT * FROM Investors WHERE Join_date<'2023-07-01';



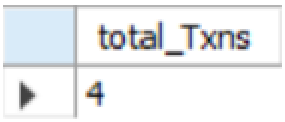
	Investor_Id	Name	Email	Phone_No	Join_Date
▶	1	Ravi Kumar	Ravi.kumar@gmail.com	9876543210	2022-01-15
	2	Senha Shah	Sneha.shah@gmail.com	9876500000	2022-05-10
	3	Amit Patel	Amit.patel@gmail.com	9123456789	2023-03-20
*	NULL	NULL	NULL	NULL	NULL

6) Aggregate Function:

a) Count Function:

Count total number of transactions.

```
SELECT COUNT(*) AS total_Txns From Transactions;
```

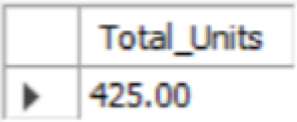


	total_Txns
▶	4

b) Sum Function:

Display The Total Units Buy From All Investors.

```
SELECT SUM(Units) AS Total_Units FROM Transactions;
```

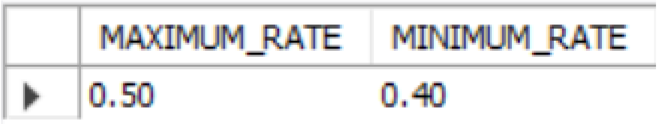


	Total_Units
▶	425.00

c) Max,Min Function:

Find The Highest And Lowest Commission rate.

```
SELECT MAX(Cummission_Rate) AS MAXIMUM_RATE,
MIN(Cummission_Rate) AS MINIMUM_RATE FROM Brokers;
```

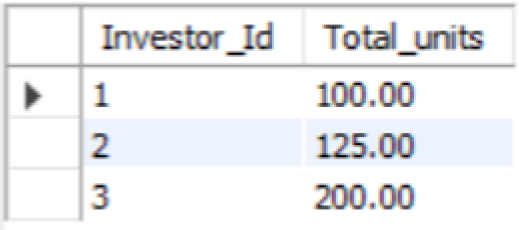


	MAXIMUM_RATE	MINIMUM_RATE
▶	0.50	0.40

7) Group by Clause:

Total Units invested by each Investor.

```
SELECT Investor_Id, SUM(Units) AS Total_units FROM Transactions GROUP BY Investor_id;
```

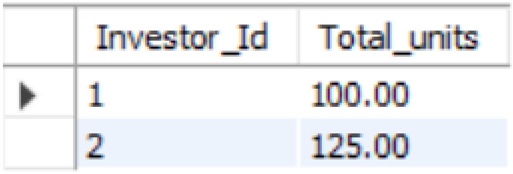


	Investor_Id	Total_units
▶	1	100.00
	2	125.00
	3	200.00

8) Having Clause:

Investors with total units > 150.

```
SELECT Investor_Id, SUM(Units) AS Total_units FROM Transactions GROUP BY Investor_id HAVING Total_units < 150;
```

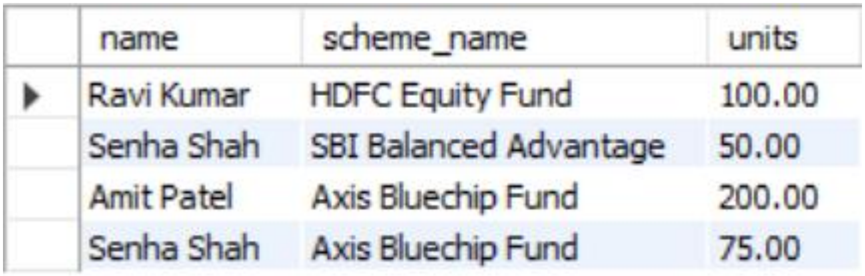


	Investor_Id	Total_units
▶	1	100.00
	2	125.00

9) Joins:

Show Investor name, Fund name and Units invested.

```
SELECT i.name, f.scheme_name, t.units FROM Investors I JOIN Transactions t ON i.investor_id = t.investor_id JOIN Funds f ON t.fund_id = f.fund_id;
```

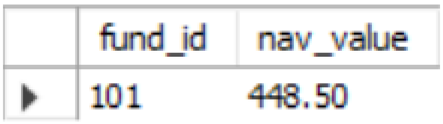


	name	scheme_name	units
▶	Ravi Kumar	HDFC Equity Fund	100.00
	Senha Shah	SBI Balanced Advantage	50.00
	Amit Patel	Axis Bluechip Fund	200.00
	Senha Shah	Axis Bluechip Fund	75.00

10) Subquery:

Find Funds with NAV higher than average NAV.

```
SELECT fund_id, nav_value FROM NAV_History WHERE nav_value >
(SELECT AVG(nav_value) FROM NAV_History);
```

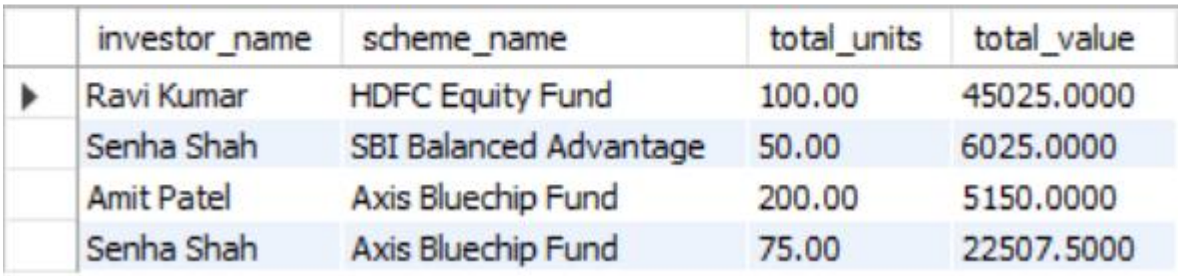


	fund_id	nav_value
▶	101	448.50

11) Views:

Investor Portfolio Summary.

```
CREATE VIEW Investor_Portfolio AS SELECT i.name AS investor_name,
f.scheme_name, SUM(t.units) AS total_units, SUM(t.units * t.nav) AS
total_value FROM Investors I JOIN Transactions t ON i.investor_id =
t.investor_id JOIN Funds f ON t.fund_id = f.fund_id GROUP BY i.name,
f.scheme_name;
```



	investor_name	scheme_name	total_units	total_value
▶	Ravi Kumar	HDFC Equity Fund	100.00	45025.0000
	Senha Shah	SBI Balanced Advantage	50.00	6025.0000
	Amit Patel	Axis Bluechip Fund	200.00	5150.0000
	Senha Shah	Axis Bluechip Fund	75.00	22507.5000