Supriya Biswas

Glasgow Clyde College

TED500

Table of Contents

[Project summary 4](#_Toc1722736)

[End Users 5](#_Toc1722738)

[Constraints 5](#_Toc1722739)

[Functional Requirements 5](#_Toc1722740)

[Non-Functional Requirements 8](#_Toc1722747)

[Use Case Diagram 9](#_Toc1722748)

[Entity Relationship Diagram 9](#_Toc1722749)

[Data Dictionary 9](#_Toc1722750)

Create Table SQL statements [9](#_Toc1722752)

[Create Table SQL statements 9](#_Toc1722751)

Insert Data Statements [9](#_Toc1722752)

Select SQL Statements [9](#_Toc1722752)

Database Testing Plan [9](#_Toc1722752)

Testing Logs [9](#_Toc1722752)

# Project Summary

TED\* 500 is a financial services provider that offers services such as:

* Investments
* Long-term bonds
* Pensions advice
* Share trading services

The company wants to build a websites that interacts with a database to allow administrators to search for shares, view share prices and process customer share investments and sales.

The proposed solution is required to perform the following queries.

* Provide list of customers.
* Add additional customers.
* Search available stocks and their unit prices by name or price.
* Provide detail of the portfolio for any specific customer.

(List of shares and their total price = Unit Price X Number of shares).

* Add new share purchases.
* Sell existing shares.

Employees at TED\*500 should be able to update available stocks and shares in database. Stock process are present in the database but are updated dynamically using a link to the London stock Exchange, this functionality will be implemented eventually, but will not be part of the prototype I have been asked to build.

The database will be used by two main types of user:

* Trading clerks
* Customers

Trading clerk will need full access to all five of the above requirements while customers will be able to view only a share portfolio and a list of available stocks.

# End Users

The database will be used by two main types of user

* Trading clerks
* Customers

# Constraints

* Ensure that column cannot have a NOT NULL value.
* Ensure that all values in a column are UNIQUE or different.
* PRIMARY KEY with the combination of not null and unique key.
* FOREIGN KEY uniquely identifies row/record in another table.
* Ensures that all values in a column satisfies a specific condition.
* Sets a default value for a column when no value is specified.

# Functional Requirements

* Provide a list of customers
* Add additional customers
* Search available stocks and their unit prices by name or price
* Provide details of the share portfolio for any specific customer

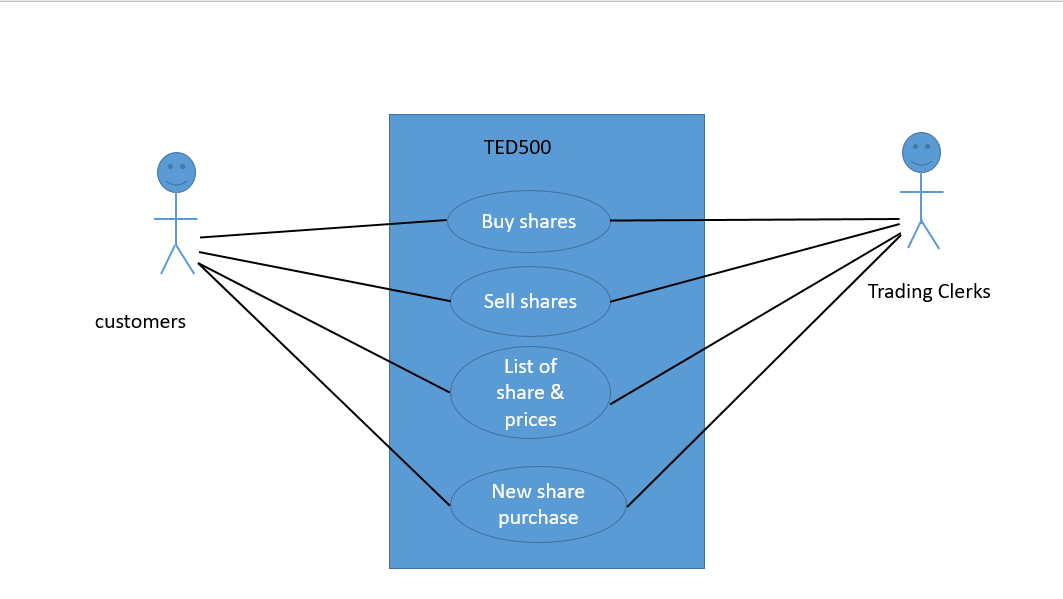
(List of shares and their total price = Unit price X Number of shares)

* Add new share purchases.
* Sell existing shares.

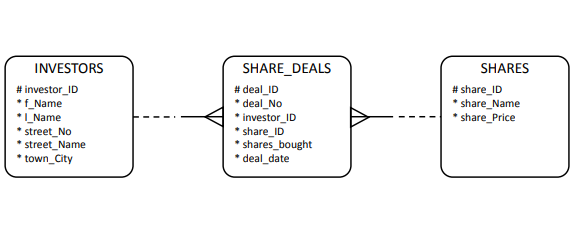
# Non-Functional Requirements

* Performance
* Scalability
* Reliability
* Maintainability
* Security
* Data Integrity
* Usability
* Interoperability

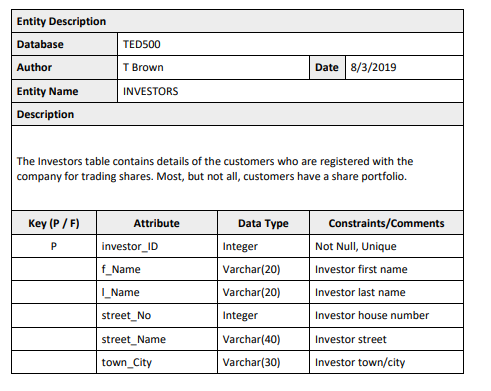
# Use Case Diagram

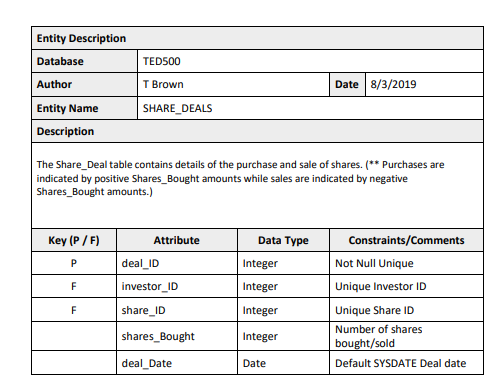


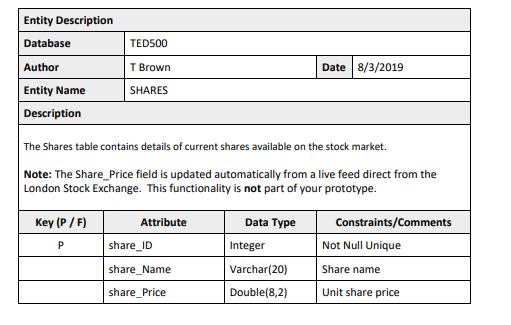
# Entity Relationship Diagram



# Data Dictionary







# Create Table SQL statements

**INVESTORS:**

CREATE TABLE INVESTORS (

investor\_ID int PRIMARY KEY AUTO\_INCREMENT,

f\_Name varchar(20),

l\_Name varchar(20),

street\_No int,

street\_Name varchar(40),

town\_City varchar(30)

)

**SHARES:**

CREATE TABLE SHARES (

share\_ID int PRIMARY KEY AUTO\_INCREMENT,

share\_Name varchar(20),

share\_Price Double(8,2)

)

**SHARE\_DEALS:**

CREATE TABLE SHARES\_DEALS (

deal\_ID int PRIMARY KEY AUTO\_INCREMENT,

shares\_Bought int,

deal\_Date TIMESTAMP DEFAULT now(),

investor\_ID int,

share\_ID int,

FOREIGN KEY fk\_investors\_ID(investor\_ID) REFERENCES INVESTORS(investor\_ID),

FOREIGN KEY fk\_share\_ID(share\_ID) REFERENCES SHARES(share\_ID)

)

# Insert Data Statements

**INVESTORS**

INSERT INTO `INVESTORS` (`investor\_ID`,`f\_Name`,`l\_Name`,`street\_No`,`street\_Name`,`town\_City`)

VALUES (1,"Dustin","Davenport",80,"Purus Street","Hines Creek"),

(2,"Norman","Dejesus",76,"5001 Sodales Street","Forres"),

(3,"Tamara","Hunt",48,"Proin St","Gallicchio"),

(4,"Deirdre","Joyce",50,"Et Av","Fourbechies"),

(5,"Charissa","Battle",27,"Integer Avenue","Camborne"),

(6,"Regan","Gillespie",49,"Ut Street","Haren"),

(7,"Murphy","Harrington",3,"Libero Street","Mendonk"),

(8,"Gannon","Durham",96,"Convallis Avenue","Bulandshahr"),

(9,"Alexis","Stark",13,"Aliquet Rd","Devon"),

(10,"Wallace","Keller",55,"Urna Rd","Les Bons Villers"),

(11,"Mikayla","Myers",74,"Sit Rd","Ternitz"),

(12,"Chaim","Carlson",80,"Nascetur Av","Comeglians"),

(13,"Oliver","Mayo",39,"Dis Rd","Trois-Rivires"),

(14,"Nelle","Torres",50,"Nisi Ave","Linares"),

(15,"Marsden","Jarvis",23,"Augue Street","Daman"),

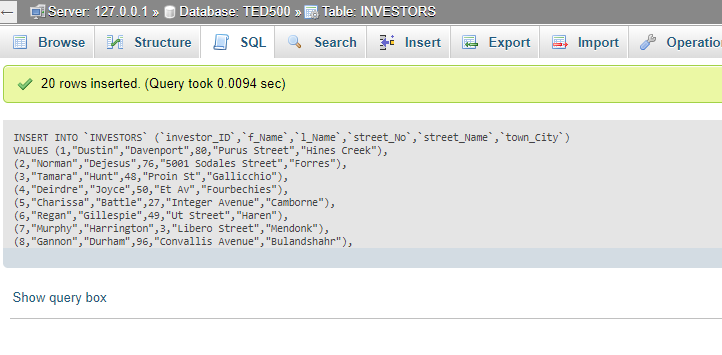
(16,"Mercedes","Norton",64,"Leo Rd","Lugo"),

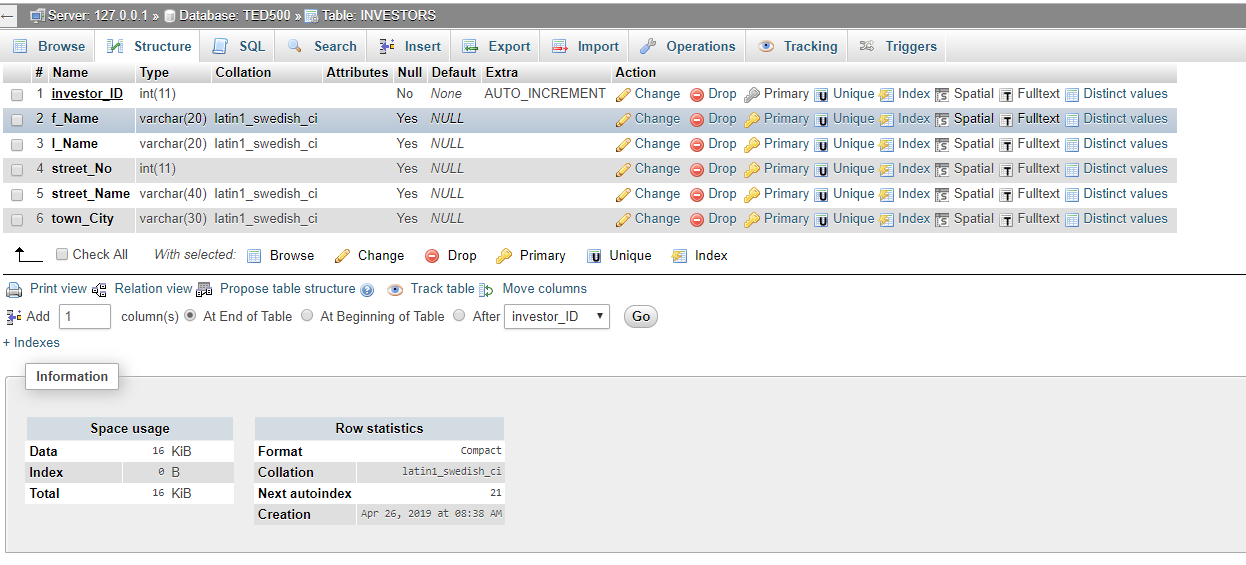
(17,"Zelenia","Sanders",48,"Morbi Av","Los Angeles"),

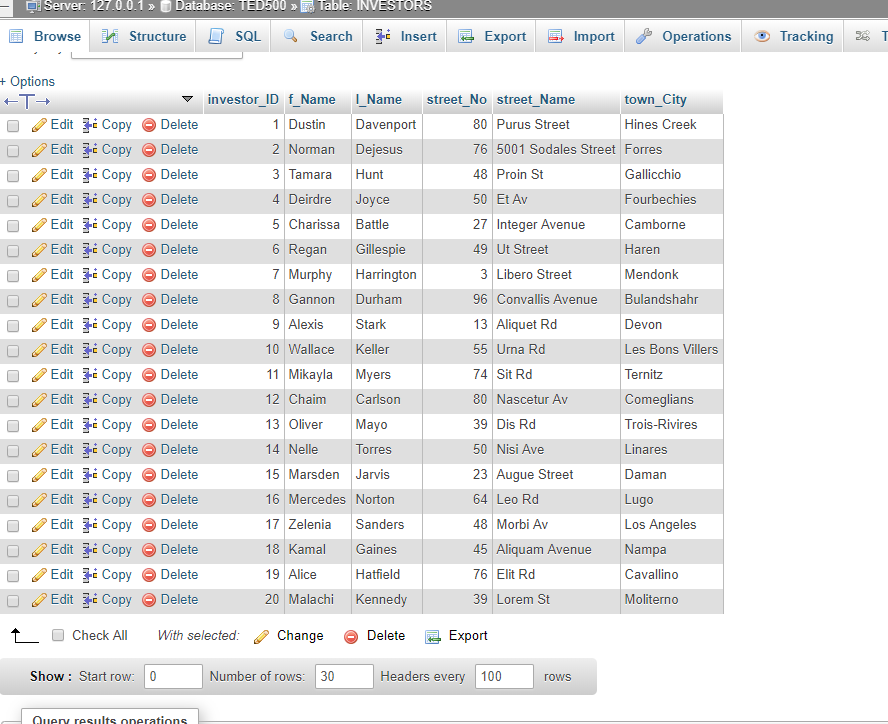
(18,"Kamal","Gaines",45,"Aliquam Avenue","Nampa"),

(19,"Alice","Hatfield",76,"Elit Rd","Cavallino"),

(20,"Malachi","Kennedy",39,"Lorem St","Moliterno");







**SHARES**

INSERT INTO `SHARES` (`share\_ID`,`share\_Name`,`share\_Price`)

VALUES (1,"Lloyds Banking Group",6.42),

(2,"Aviva plc",2.41),

(3,"Sirius Minerals plc",1.51),

(4,"Scottish Mortgage",4.32),

(5,"ASOS plc",2.35),

(6,"BAE Systems plc",4.87),

(7,"Widecells Group plc",6.04),

(8,"Vodafone Group plc",2.67),

(9,"GVC Holdings plc",2.64),

(10,"Kier Group plc",2.30),

(11,"Debenhams plc",1.27),

(12,"Cadence Minerals",9.45),

(13,"BP Plc",7.71),

(14,"Barclays plc",1.15),

(15,"Hurricane Energy",2.72),

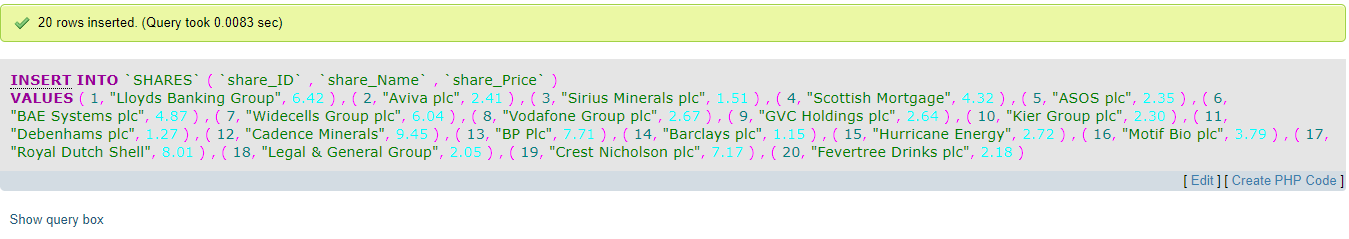
(16,"Motif Bio plc",3.79),

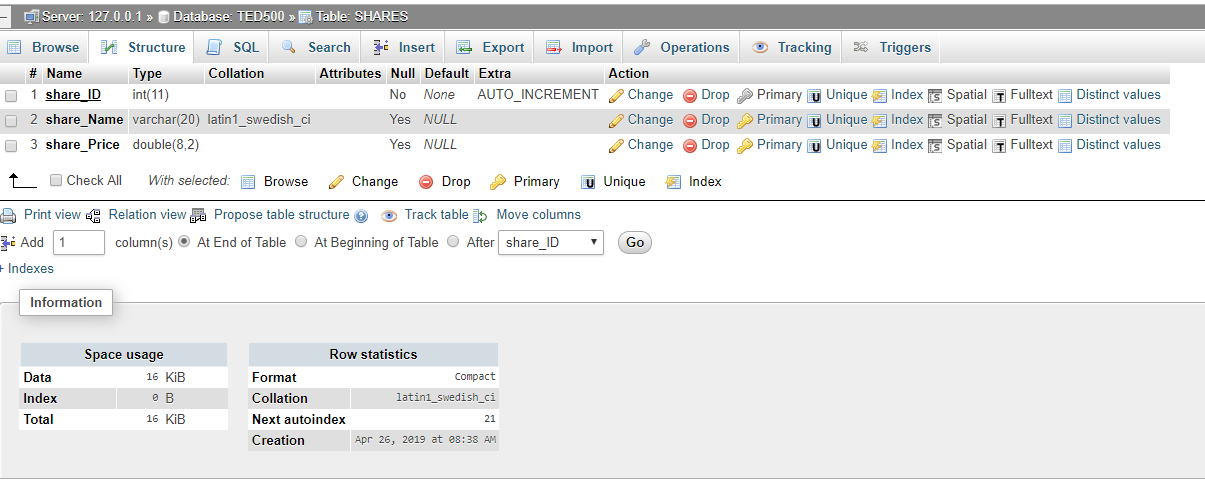
(17,"Royal Dutch Shell",8.01),

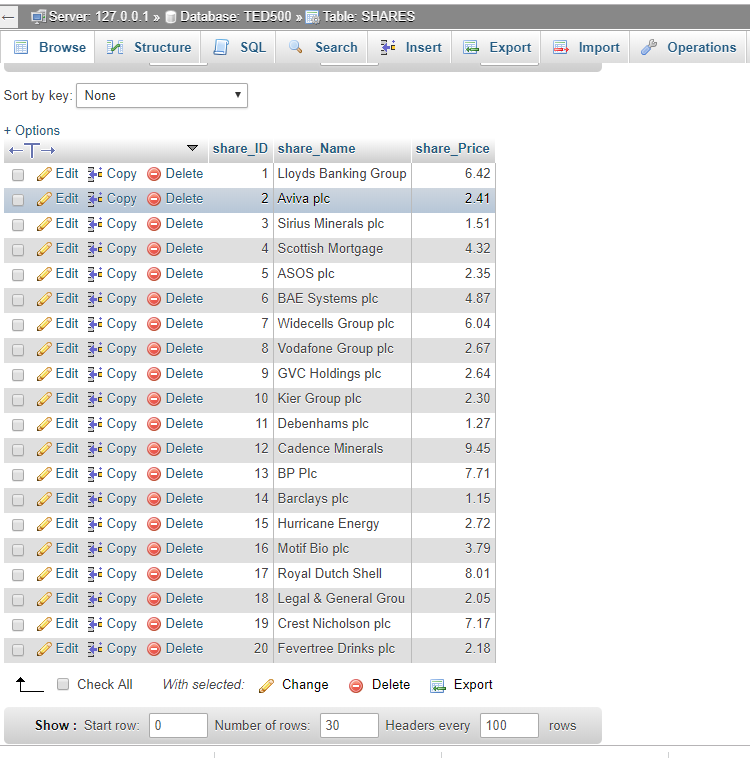
(18,"Legal & General Group",2.05),

(19,"Crest Nicholson plc",7.17),

(20,"Fevertree Drinks plc",2.18);







**SHARES\_DEALS**

INSERT INTO `SHARE\_DEALS` (`deal\_ID`,`investor\_ID`,`share\_ID`,`shares\_Bought`,`deal\_Date`)

VALUES (1,5,9,72,"2018-11-04 01:41:44"),

(2,14,4,69,"2018-08-22 06:38:56"),

(3,2,20,60,"2018-10-14 07:19:06"),

(4,19,15,112,"2018-10-17 15:21:18"),

(5,20,6,163,"2018-08-13 17:37:19"),

(6,2,1,177,"2018-08-29 22:58:31"),

(7,3,16,92,"2018-05-18 10:16:18"),

(8,13,12,95,"2018-07-20 04:25:13"),

(9,9,19,52,"2018-09-21 06:11:52"),

(10,18,17,40,"2018-04-07 21:41:37"),

(11,2,15,199,"2018-07-21 06:33:58"),

(12,12,20,15,"2018-11-19 05:42:57"),

(13,19,7,163,"2018-10-31 10:34:06"),

(14,13,7,191,"2018-08-09 23:36:16"),

(15,4,12,170,"2018-07-02 02:23:52"),

(16,10,1,172,"2018-06-27 00:04:26"),

(17,12,20,68,"2018-11-23 03:55:40"),

(18,10,5,77,"2018-10-04 21:18:24"),

(19,11,16,120,"2018-12-06 03:11:39"),

(20,9,19,101,"2018-10-22 05:22:30"),

(21,20,2,8,"2018-11-20 05:47:23"),

(22,2,4,154,"2018-12-08 02:37:48"),

(23,8,5,156,"2018-10-14 05:33:52"),

(24,8,10,164,"2018-06-13 16:57:59"),

(25,1,19,139,"2018-12-02 00:33:33"),

(26,20,12,54,"2018-05-12 01:34:56"),

(27,6,19,161,"2018-06-17 15:46:42"),

(28,1,11,183,"2018-11-26 13:45:05"),

(29,10,8,120,"2018-11-12 19:54:15"),

(30,20,2,151,"2018-11-26 22:42:53"),

(31,10,19,23,"2018-05-08 09:01:51"),

(32,7,15,132,"2018-06-07 04:20:50"),

(33,11,9,164,"2018-11-09 02:31:23"),

(34,18,17,150,"2018-06-13 23:45:20"),

(35,16,6,148,"2018-07-27 08:11:45"),

(36,8,16,110,"2018-07-09 02:09:31"),

(37,4,17,183,"2018-11-10 05:58:04"),

(38,1,2,145,"2018-09-23 04:50:52"),

(39,5,10,23,"2018-08-26 02:21:17"),

(40,19,2,89,"2018-08-16 14:35:43"),

(41,11,19,53,"2018-08-25 20:23:53"),

(42,13,20,174,"2018-03-29 11:23:29"),

(43,12,13,24,"2018-06-08 05:26:21"),

(44,18,5,42,"2018-09-13 10:38:28"),

(45,2,10,5,"2018-04-14 17:59:52"),

(46,9,5,9,"2018-12-06 07:50:51"),

(47,17,9,9,"2018-07-09 05:12:22"),

(48,16,7,163,"2018-07-01 05:52:26"),

(49,16,4,45,"2018-11-08 04:35:53"),

(50,3,19,129,"2018-10-05 14:16:14"),

(51,17,1,85,"2018-08-16 19:32:08"),

(52,1,11,16,"2018-03-30 20:13:05"),

(53,18,13,121,"2018-08-23 02:17:32"),

(54,5,9,1,"2018-04-24 16:28:25"),

(55,18,12,44,"2018-05-24 00:58:49"),

(56,11,14,80,"2018-10-23 21:44:56"),

(57,17,8,190,"2018-06-14 02:47:14"),

(58,4,4,170,"2018-09-12 17:59:54"),

(59,8,13,21,"2018-04-18 02:11:41"),

(60,5,15,87,"2018-04-15 14:21:11"),

(61,17,6,17,"2018-11-25 14:47:14"),

(62,17,13,54,"2018-06-28 19:09:08"),

(63,1,7,63,"2018-04-12 04:43:24"),

(64,15,8,147,"2018-07-13 09:47:09"),

(65,9,4,34,"2018-06-17 08:48:55"),

(66,1,5,127,"2018-06-03 21:19:17"),

(67,4,16,37,"2018-12-16 23:57:10"),

(68,1,15,84,"2018-10-01 06:27:44"),

(69,12,20,182,"2018-10-29 20:10:13"),

(70,17,6,125,"2018-12-26 06:19:58"),

(71,13,20,178,"2018-11-07 02:40:26"),

(72,10,17,102,"2018-07-28 12:56:23"),

(73,9,16,168,"2018-10-29 04:41:00"),

(74,17,16,160,"2018-04-15 17:21:54"),

(75,6,1,69,"2018-08-28 10:42:47"),

(76,3,7,198,"2018-10-24 01:52:52"),

(77,7,20,111,"2018-06-18 01:33:30"),

(78,19,10,191,"2018-10-26 16:02:19"),

(79,8,14,75,"2018-10-23 03:02:13"),

(80,12,17,114,"2018-12-31 17:47:14"),

(81,19,13,195,"2018-08-23 17:01:18"),

(82,19,8,135,"2018-11-17 21:40:12"),

(83,11,19,197,"2018-06-14 17:51:32"),

(84,20,5,97,"2018-09-25 00:33:48"),

(85,12,4,84,"2018-12-16 17:40:39"),

(86,6,19,184,"2018-10-08 11:47:50"),

(87,19,1,37,"2018-11-30 08:50:56"),

(88,13,19,124,"2018-08-10 05:03:04"),

(89,3,14,44,"2018-10-03 23:31:14"),

(90,16,12,124,"2018-09-06 07:40:34"),

(91,10,12,29,"2018-05-10 00:21:50"),

(92,20,3,43,"2018-07-15 15:42:27"),

(93,9,3,149,"2018-03-29 10:39:48"),

(94,8,10,177,"2018-12-23 10:44:24"),

(95,18,19,74,"2018-07-14 19:22:37"),

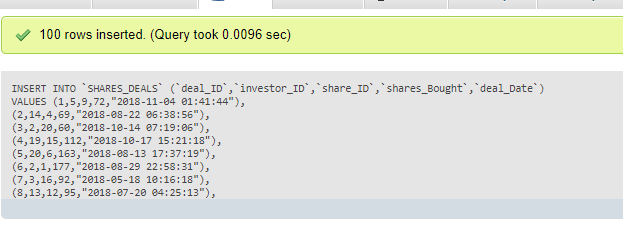
(96,7,8,92,"2018-06-08 16:33:25"),

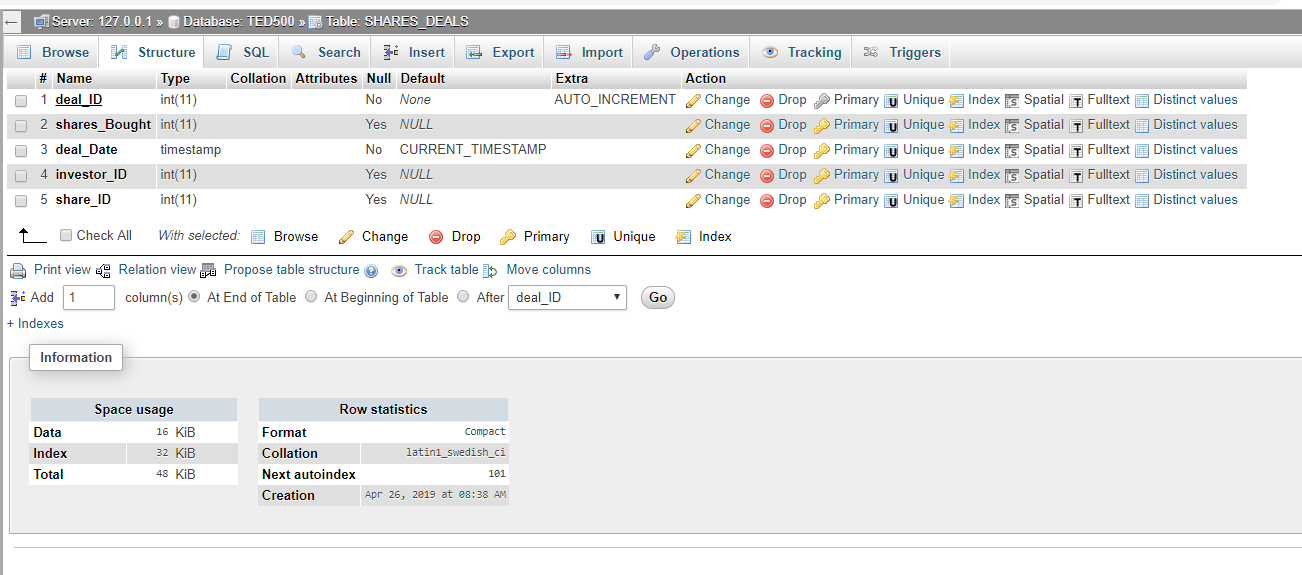
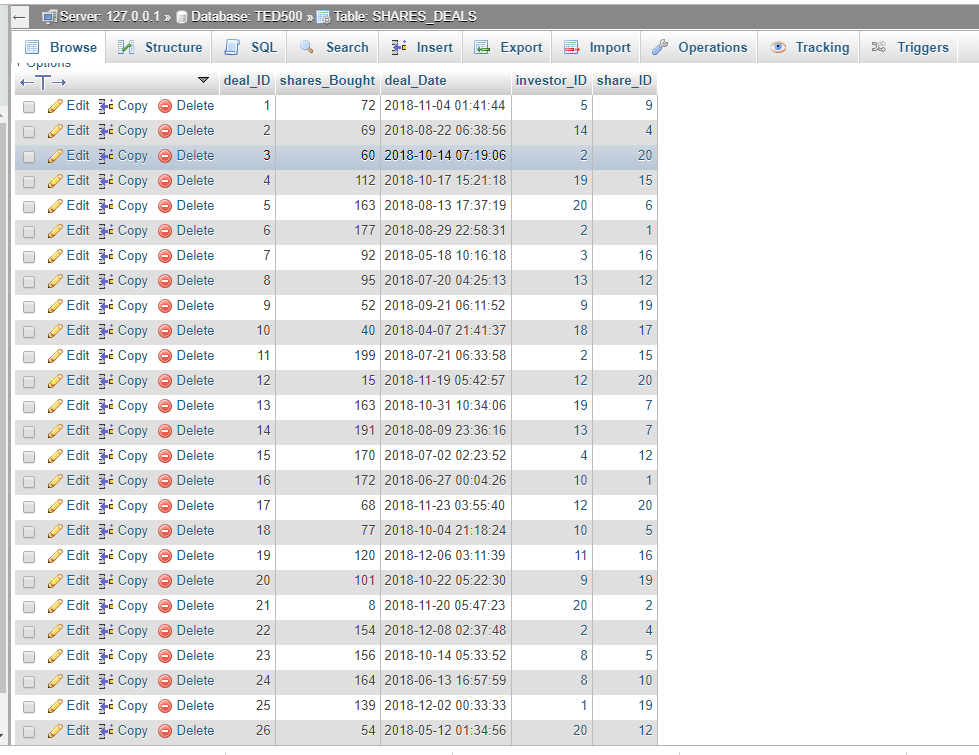
(97,19,6,65,"2018-07-31 17:33:13"),

(98,13,18,150,"2018-11-28 00:58:17"),

(99,13,4,16,"2018-05-05 00:50:47"),

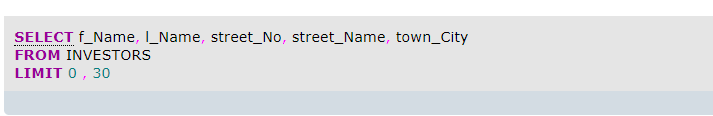
(100,10,7,103,"2018-12-16 06:28:03");





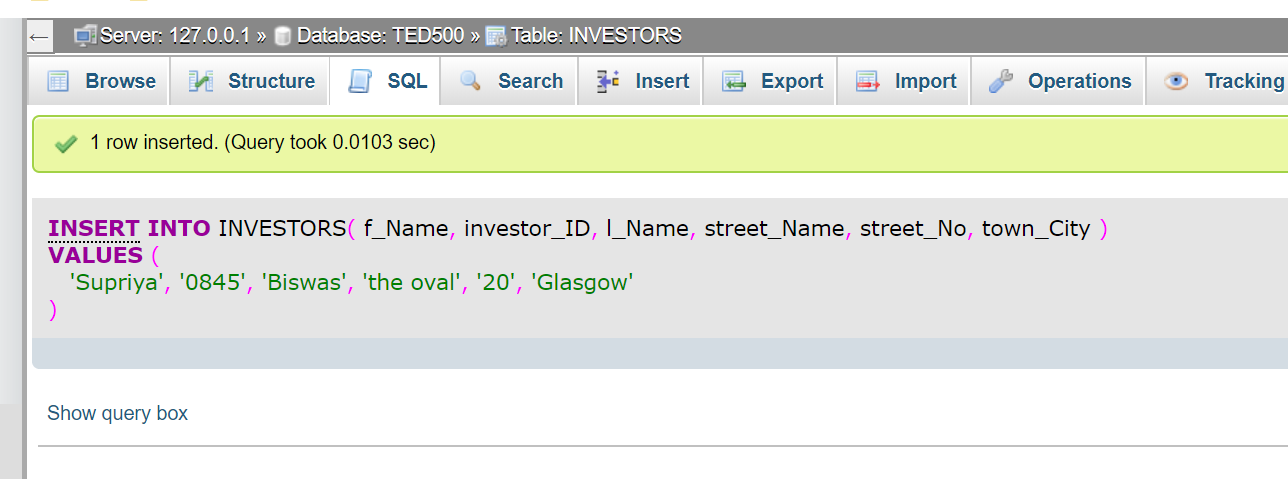
# Select SQL statements

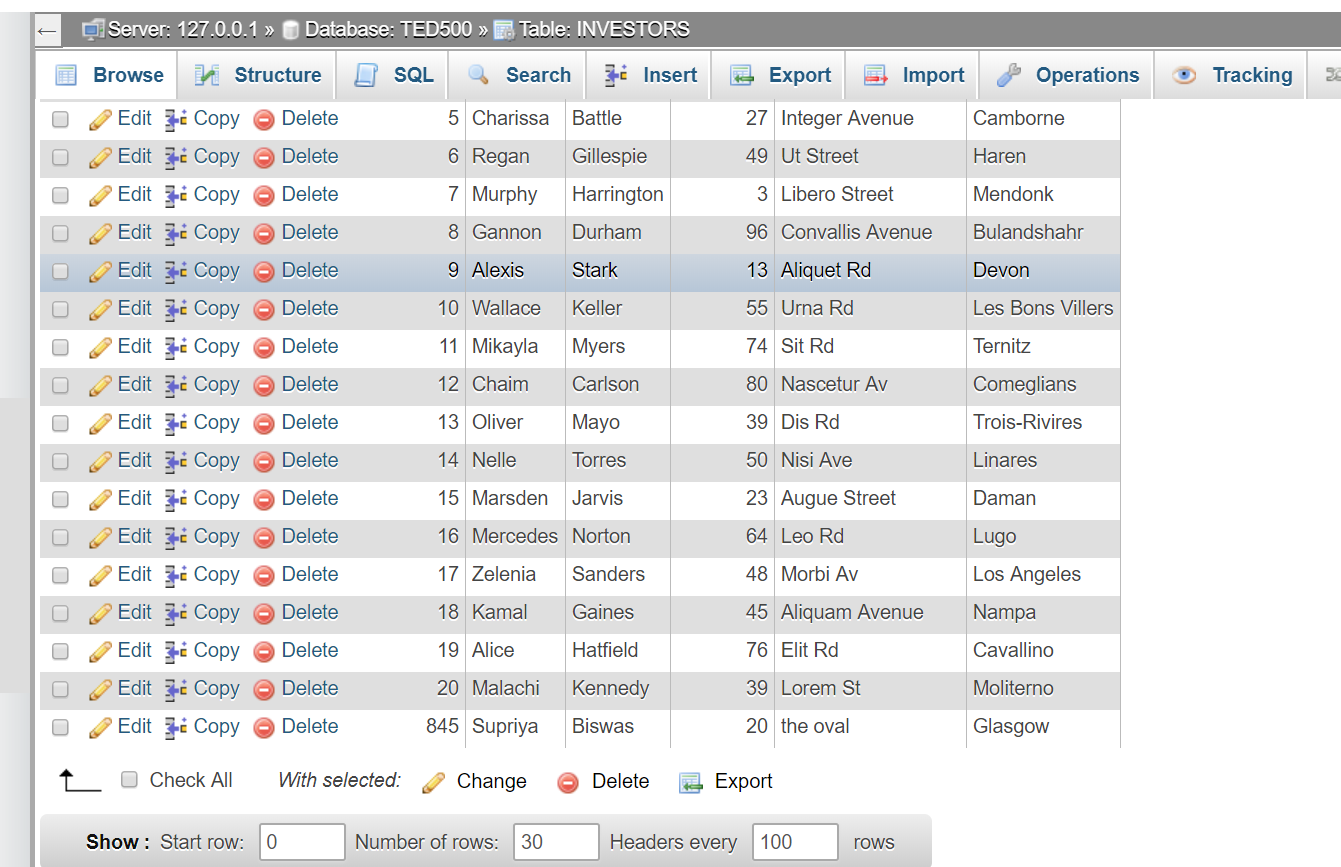
1. Provide a list of customer names and addresses.



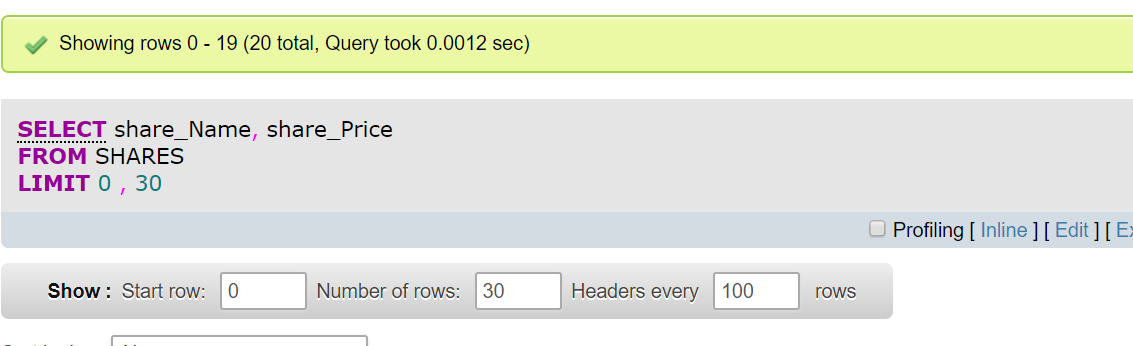


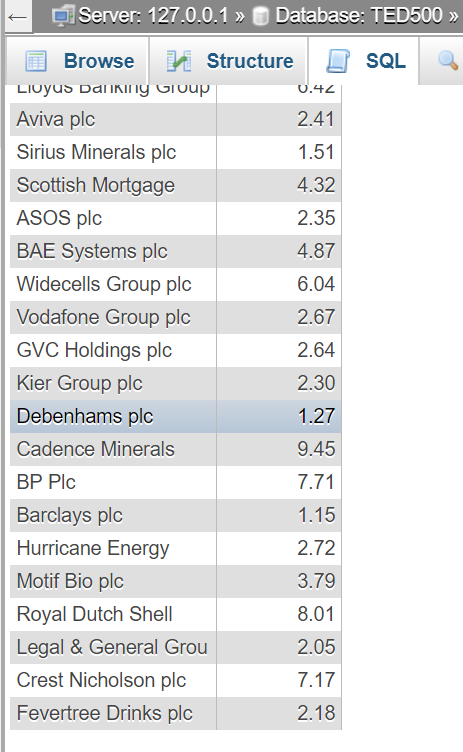
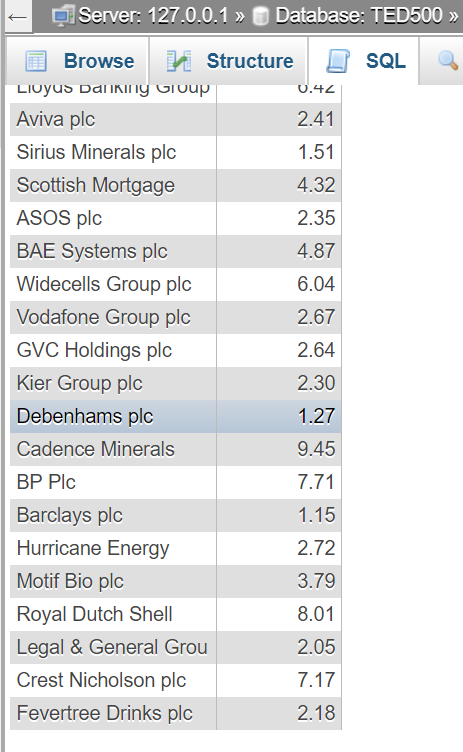
1. Add additional customers.



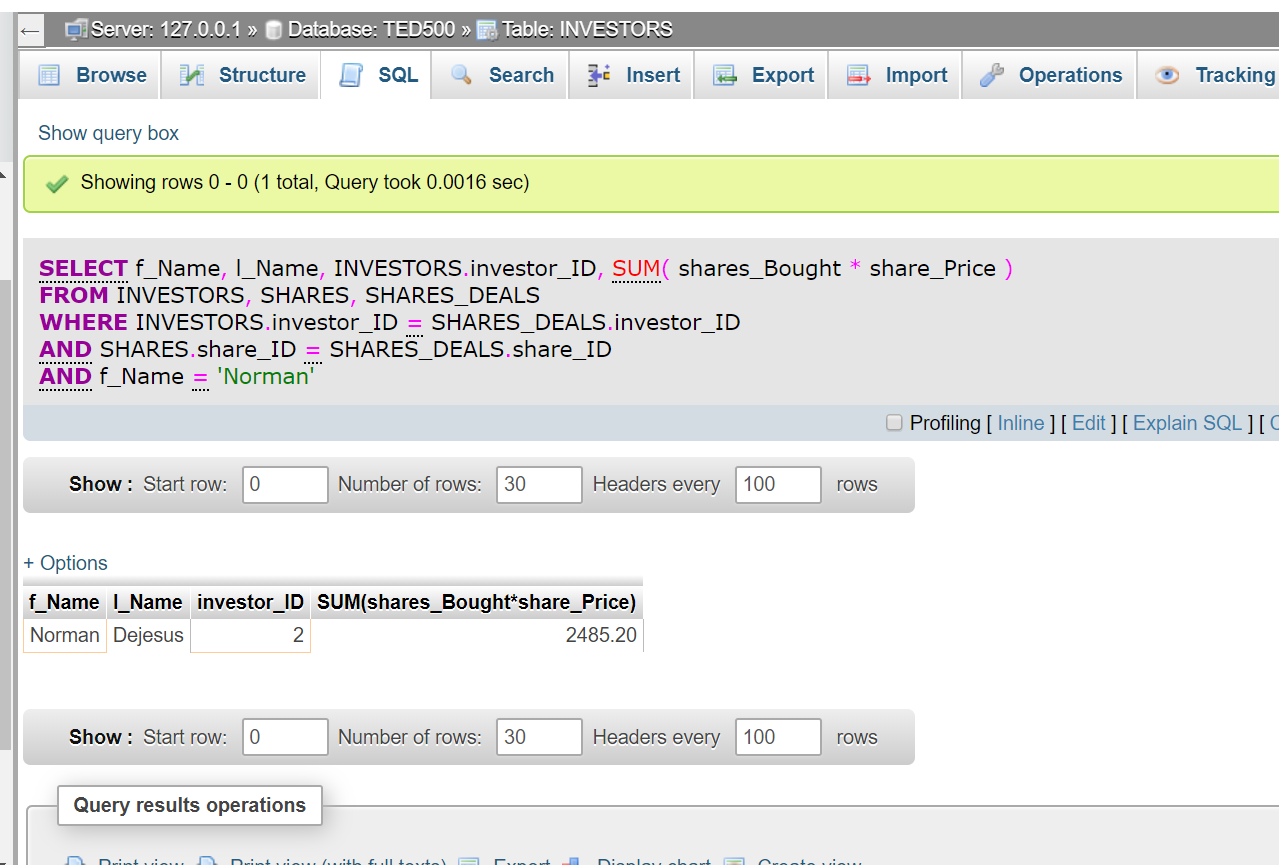


1. ) Search available shares and their unit prices by price-range listing the share name and price in each case.





1. Provide details of the share portfolio for any specific customer listing the customer name, ID, share deals made and total value of each deal.



1. Add new share deal purchases and sales.

CREATE TABLE SHARES\_SALES\_DEALS (

deal\_ID int PRIMARY KEY AUTO\_INCREMENT,

shares\_sold int,

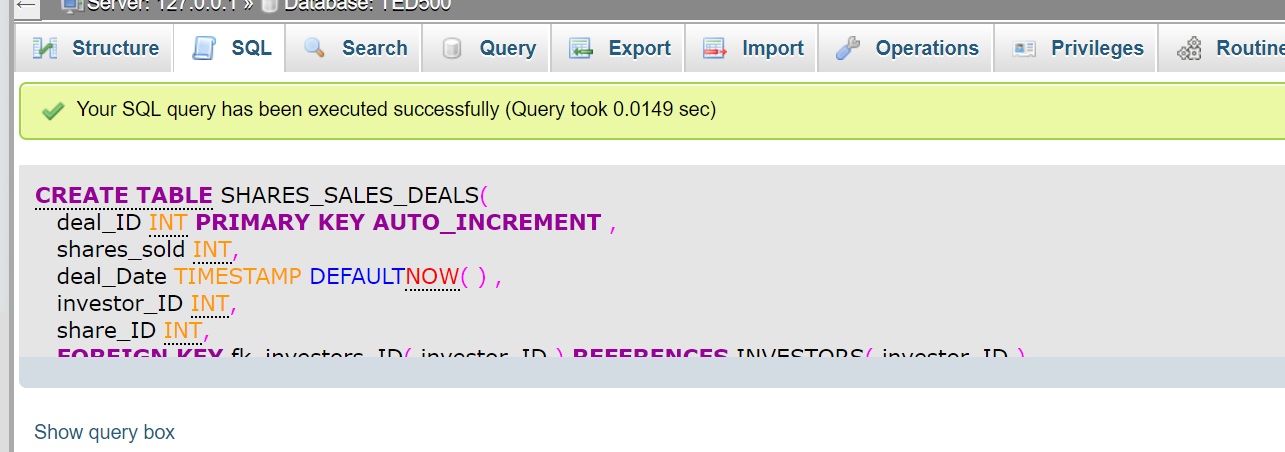
deal\_Date TIMESTAMP DEFAULT now(),

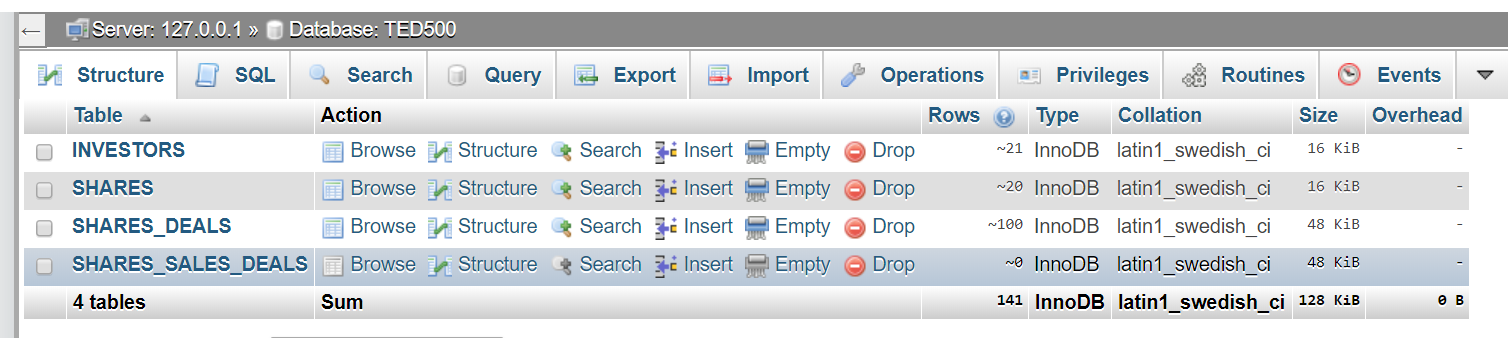
investor\_ID int,

share\_ID int,

FOREIGN KEY fk\_investors\_ID(investor\_ID) REFERENCES INVESTORS(investor\_ID),

FOREIGN KEY fk\_share\_ID(share\_ID) REFERENCES SHARES(share\_ID)

)



1. Sell existing shares. [The presence of these can be manually checked first.]

SELECT f\_name, sn.share\_name, sd.investor\_id, sn.share\_id, sum(shares\_Bought) as total\_shares

FROM `SHARES\_DEALS` sd, `SHARES` sn, `INVESTORS` i

WHERE sd.share\_ID =9

AND sd.investor\_ID =5

AND sd.share\_ID = sn.share\_ID

AND i.investor\_ID = sd.investor\_ID

group by f\_name, sn.share\_name

