

Intro to Database

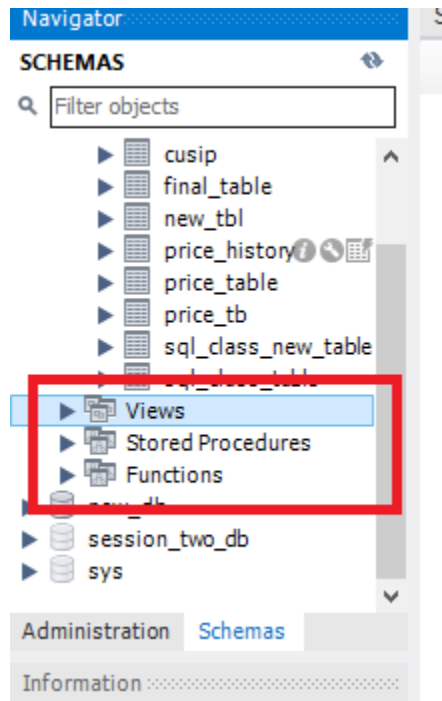
Danny Tan

11/14/2019

SQL View Table

- Result set of a stored query on the data
- Allow users to query as they would in a persistent database collection object
- Does not form part of the physical schema
- Virtual Table

View Table



```
1 • CREATE VIEW `tttt` AS
2
3 SELECT
4     `price_table`.`Date` AS `Date`,
5     `price_table`.`Open` AS `Open`,
6     `price_table`.`High` AS `High`,
7     `price_table`.`Low` AS `Low`,
8     `price_table`.`Close` AS `Close`,
9     `price_table`.`Adj Close` AS `Adj Close`,
10    `price_table`.`Volume` AS `Volume`,
11    (`price_table`.`Adj Close` * 2) AS `new_price`
12 FROM
13     `price_table`
```

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

Online DDL

Algorithm:

Default

Lock Type:

Default

```
1  USE `my_db`;
2  CREATE OR REPLACE VIEW `tttt` AS
3
4  SELECT
5      `price_table`.`Date` AS `Date`,
6      `price_table`.`Open` AS `Open`,
7      `price_table`.`High` AS `High`,
8      `price_table`.`Low` AS `Low`,
9      `price_table`.`Close` AS `Close`,
10     `price_table`.`Adj Close` AS `Adj Close`,
11     `price_table`.`Volume` AS `Volume`,
12     (`price_table`.`Adj Close` * 2) AS `new_price`
13 FROM
14     `price_table`;
```

Back

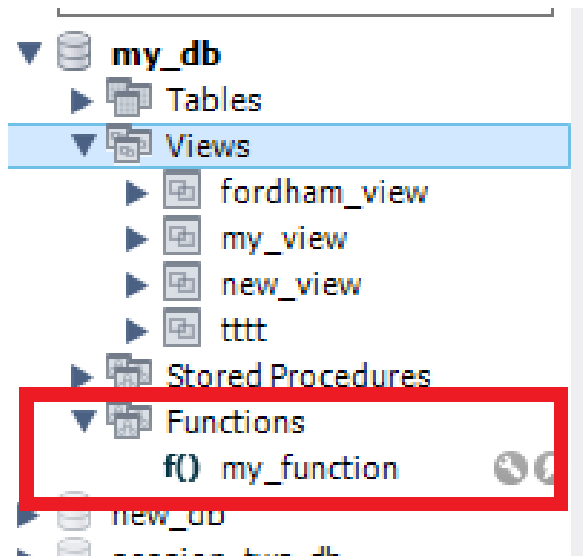
Apply

Cancel

1 • `select * from tttt`

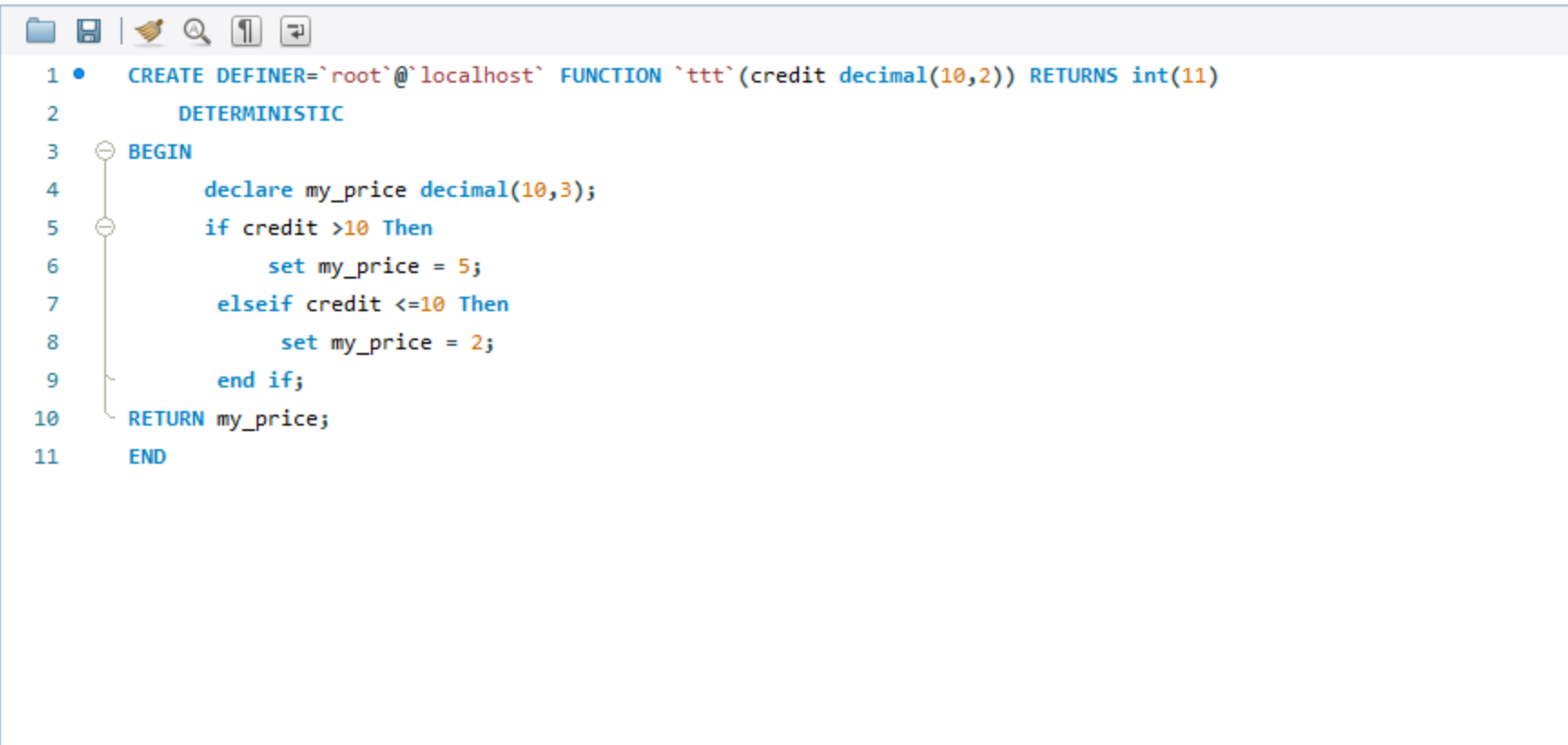
Result Grid								
Filter Rows: <input type="text"/>								
Export: Wrap Cell Content: Fetch rows:								
	Date	Open	High	Low	Close	Adj Close	Volume	new_price
▶	2018-09-11	218.009995	224.300003	216.559998	223.850006	220.429443	35749000	440.858886
	2018-09-12	224.940002	225	219.839996	221.070007	217.691925	49278700	435.38385
	2018-09-13	223.520004	228.350006	222.570007	226.410004	222.950317	41706400	445.900634
	2018-09-14	225.75	226.839996	222.520004	223.839996	220.419571	31999300	440.839142
	2018-09-17	222.149994	222.949997	217.270004	217.880005	214.550659	37195100	429.101318
	2018-09-18	217.789993	221.850006	217.119995	218.240005	214.905167	31571700	429.810334
	2018-09-19	218.5	219.619995	215.300003	218.369995	215.033142	27123800	430.066284
	2018-09-20	220.240005	222.279999	219.149994	220.029999	216.667816	26608800	433.335632
	2018-09-21	220.779999	221.360001	217.289993	217.660004	214.33403	96246700	428.66806
	2018-09-24	216.820007	221.259995	216.630005	220.789993	217.416183	27693400	434.832366

Function



Function Example

XL:



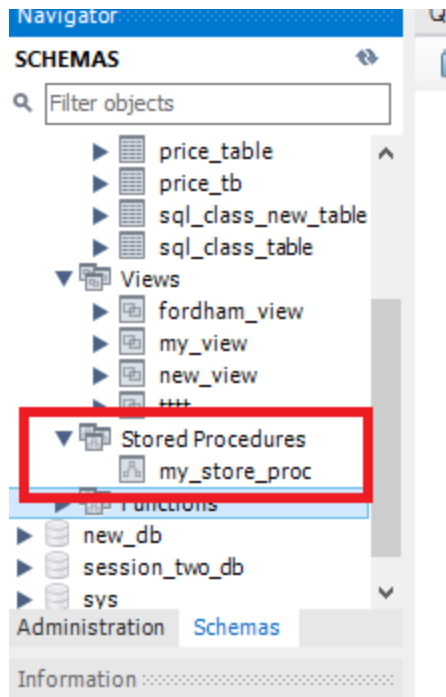
```
1 • CREATE DEFINER=`root`@`localhost` FUNCTION `ttt`(credit decimal(10,2)) RETURNS int(11)
2     DETERMINISTIC
3     BEGIN
4         declare my_price decimal(10,3);
5         if credit >10 Then
6             set my_price = 5;
7         elseif credit <=10 Then
8             set my_price = 2;
9         end if;
10    RETURN my_price;
11    END
```

The image shows a screenshot of a MySQL IDE window. The title bar is light gray with standard icons (file explorer, save, undo, redo, search, and a refresh icon). The main area is white and contains SQL code for creating a function named 'ttt'. The code is syntax-highlighted: keywords are blue, identifiers are black, and numeric literals are orange. Line numbers 1 through 11 are on the left. A vertical line with circular markers is positioned between lines 3 and 10, indicating a code block or function body. The code defines a function 'ttt' that takes a 'credit' parameter of type 'decimal(10,2)' and returns an 'int(11)'. It is marked as 'DETERMINISTIC'. The function body starts with 'BEGIN', declares a local variable 'my_price' of type 'decimal(10,3)', and then uses an 'if-elseif' statement to set 'my_price' to 5 if 'credit' is greater than 10, or 2 if 'credit' is less than or equal to 10. Finally, it returns 'my_price' and ends with 'END'.

Function Example

The screenshot shows the DBeaver SQL editor interface. At the top, there is a toolbar with various icons for file operations, execution, and navigation. Below the toolbar, a SQL query is entered in the editor: `select ttt(Open) from aap1`. The query is highlighted in blue. Below the query editor, there is a section for the query results. The results are displayed in a table with one column labeled `ttt(Open)`. The table contains 10 rows, each with the value `5`. The first row is highlighted in blue. The interface also includes a sidebar on the left with a tree view showing the database structure, and a bottom toolbar with options like `Result Grid`, `Filter Rows`, `Export`, `Wrap Cell Content`, and `Fetch rows`.

Stored Procedure



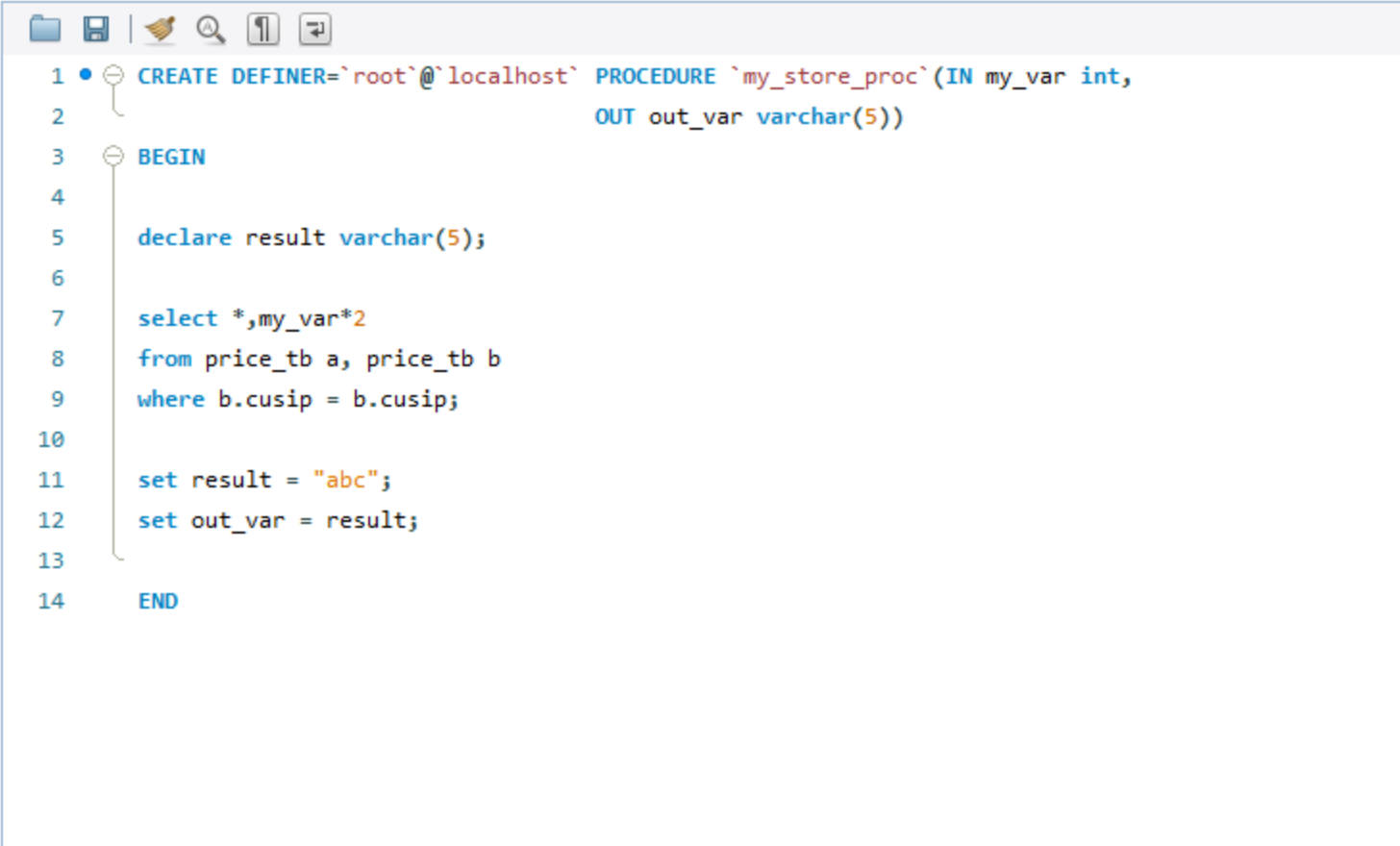
Stored Procedure

- Subroutine
- Perform collections of SQL
- Stored in DB

Stored Procedure

me: my_store_proc

DDL:



```
1 • CREATE DEFINER=`root`@`localhost` PROCEDURE `my_store_proc`(IN my_var int,
2                                OUT out_var varchar(5))
3 BEGIN
4
5     declare result varchar(5);
6
7     select *,my_var*2
8     from price_tb a, price_tb b
9     where b.cusip = b.cusip;
10
11     set result = "abc";
12     set out_var = result;
13
14     END
```

Stored Procedure

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 10 rows' dropdown. The query editor contains the text: `1 • call my_store_proc(42, @dfe)`. Below the editor, the 'Result Grid' tab is active, displaying a table with 7 columns: `trade_dt`, `cusip`, `close_price`, `trade_dt`, `cusip`, `close_price`, and `my_var*2`. The first row of data shows values for Apple stock on 2010-08-22, with a calculated value of 84 for `my_var*2`.

trade_dt	cusip	close_price	trade_dt	cusip	close_price	my_var*2
2010-08-22	aapl	100	2010-08-22	aapl	100	84