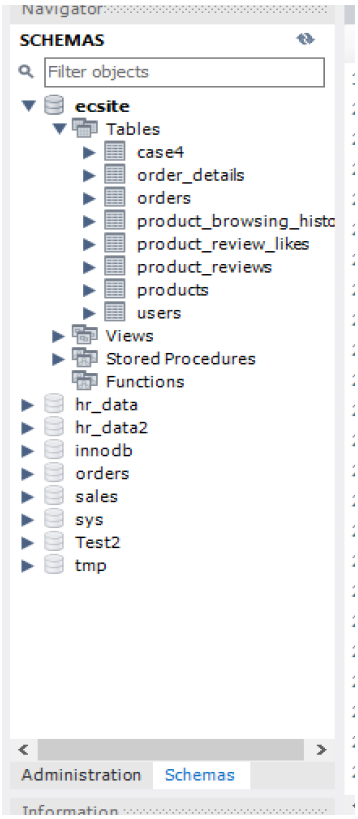
セクション 15–ハンズオンスクリプト

■ストアドプロシージャの活用

先ほどのコマンドを実行してみます。

```
DELIMITER $$
CREATE PROCEDURE avg_price()
BEGIN
	SELECT
		AVG(price)
	FROM products;
END$$
DELIMITER ;
```

と入力して、実行してみます。すると実行ができたのですが、特に処理は行われません。この時点ではプロシージャがサーバーに登録されただけとなっています。登録状況を確認してみましょう。



右タブにある SCHEMAS を更新すると、Stored Procedures の中に、先ほど作成した avg_price というプロシージャが格納されていることがわかります。このように登録されているわけです。

次に登録したプロシージャを呼び出してみましょう。

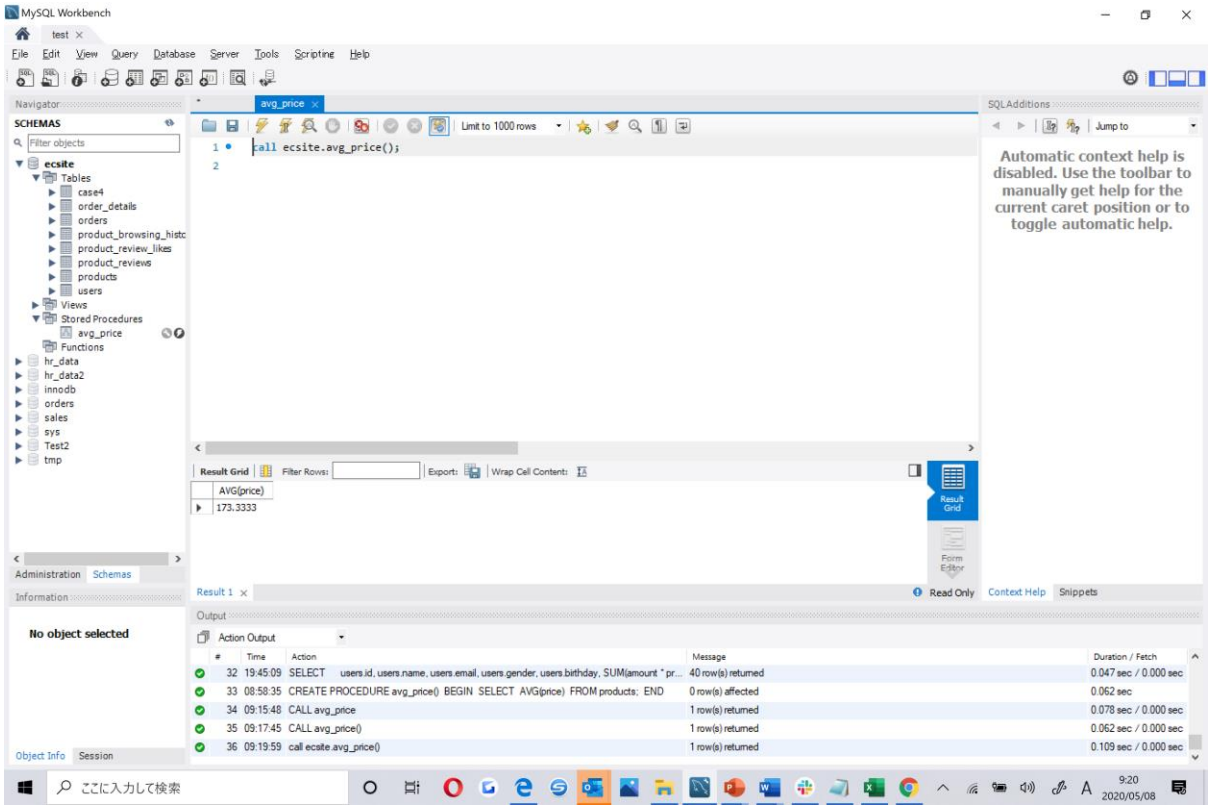
CALL avg_price;

で実行すると、先ほどの製品の平均価格を求めるクエリが実行されて、平均価格が算出されたことがわかります。このように事前にクエリを保存しておいて、呼び出すだけで実行可能にする処理がストアドプロシージャとなります。

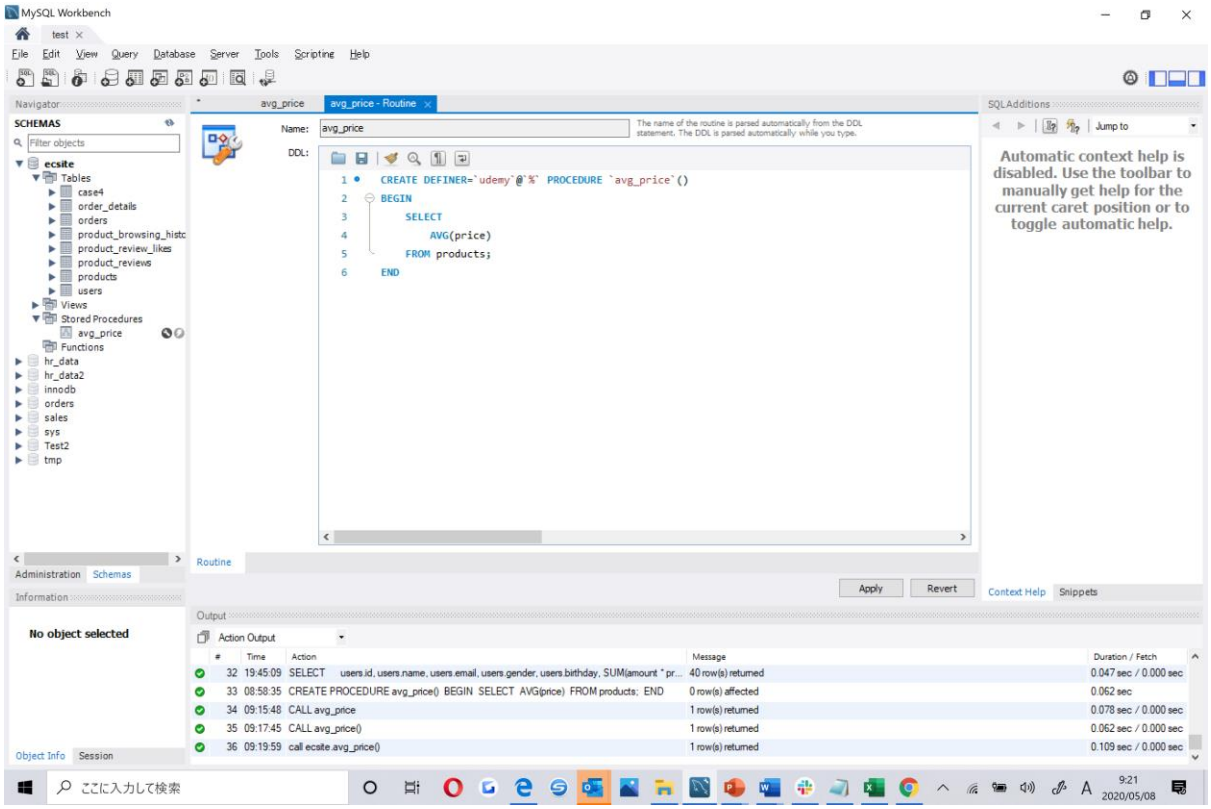
CALL avg_price();

とカッコをつけて呼び出すこともできます。結果は同じです。プロシージャの設定にパラメーターを設定することができ、これは（）にそれを設定して呼び出す際に利用される形式です。パラメーターが設定されていないプロシージャについては、カッコがある CALL とない CALL では同じ結果となります。

もう少し WORKBENCH をみていきましょう。左タブの Stored Procedures にいっていただいて、avg_price の横にある雷マークをクリックすると、同じくストアドプロシージャを実行しつつ、その呼び出しクエリが表示されます。



またレンチマークをクリックすると。ストアドプロシージャに設定されているコマンドが表示されて、ここでコマンドを編集することが可能となります。実際に編集したら、下の APPLY を押すことで、プロシージャの内容を更新することができます。



最後にこのプロシージャを削除します。

DROP PROCEDURE avg_price;

と入力して、実行することで削除することが可能です。

WORKBENCH 操作で削除する方法もやってみます。

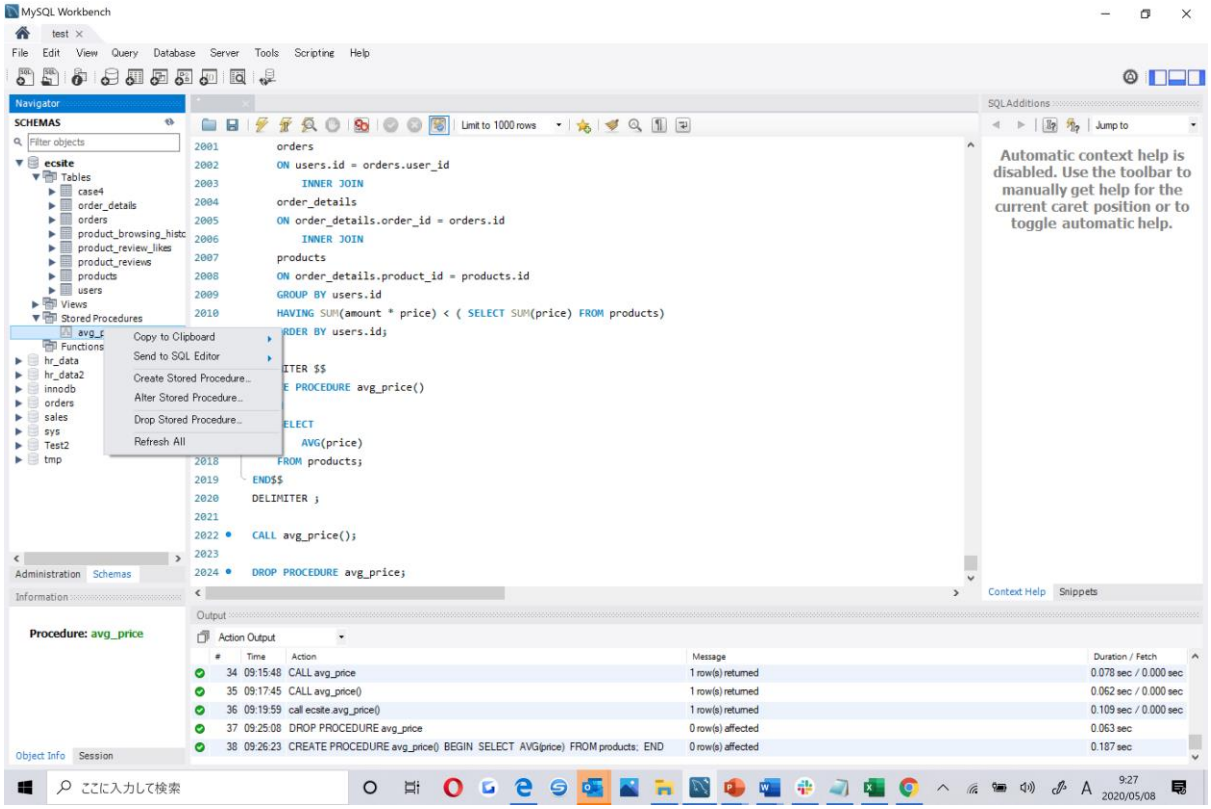
再び

```
DELIMITER $$
CREATE PROCEDURE avg_price()
BEGIN
	SELECT
		AVG(price)
	FROM products;
END$$
```

DELIMITER ;

を実行してから、

左タブのスキーマを更新して、 ストアドプロシージャの avg_price にカーソルをあわせて右クリックを押します。すると DROP STORED　PROCEDURE がでてきますので、これをクリックします。DROP NOW を押していただければ削除されます。



このように WORKBENCH 上の操作でも作成したり、更新したり、削除したりすることが可能です。

■パラメーターの活用

こちらのコマンドを確認していきます。

```
DELIMITER $$
CREATE PROCEDURE users_info(IN input INT)
BEGIN
        SELECT * FROM users
    WHERE id = input;
END$$
DELIMITER ;
```

と入力して、実行します。
このプロシージャでは、インプットパラメーターとして、input を INT 型で定義しています。その上で、WHERE 句の条件として、INPUT に入力した値を users_id の条件として利用して、ユーザーを抽出するという設定を行っています。つまり、ユーザーID をインプットにして、そのユーザーの個人情報を抽出するプロシージャとなるわけです。

これでプロシージャが作成されたので、今度は実行してみます。

```
CALL users_info(5);
```

というように、プロシージャ名のあとの（）において、インプットを設定して呼び出すことで実行することができます。この結果では、ID５のユーザーのデータが抽出されました。

次に Workbench 操作でも実行してみましょう。WORKBENCH の左タブを更新していただいて、先ほどつくったプロシージャを表示します。その上で、users_info にカーソルを合わせていただいて、雷マークを押していただくと、インプットを入力する画面がポップアップで表示されます。ここに数値をいれて実行すると、そのユーザーのデータが抽出されます。

このように GUI 上でシステム操作のように実行することができます。

次に、OUTPUT 値の設定を加えていきます。一旦先ほどのプロシージャ―を削除します。

```
DROP PROCEDURE users_info;
```

と入力して、実行します。

次に先ほどのプロシージャに OUTPUT を追加していきます。IN のときと同じように、今度は OUT と指定してから output 名称と、データ型を指定します。OUT output varchar(255)と指定していきましょう。

次に SELECT　アウトプットとして設定したいカラム名称　INTO　output 名称と入力していきます。SELECT email INTO output としていきましょう。

```
DELIMITER $$
CREATE PROCEDURE users_info(IN input int, OUT output varchar(255) )
BEGIN
        SELECT email INTO output
    FROM users
    WHERE id = input;
END$$
DELIMITER ;
```

これで実行します。プロシージャが作成されました。呼び出してみます。

```
call users_info(1, @output);
```

と実行すると OUTPUT が設定されます。これだけでは表示されないため、次に SELECT 文で呼び出します。

```
SELECT @output;
```

と入力して、実行すると指定した ID のユーザーの email がアウトプットとして表示されました。

次に workbench で実行してみましょう。右のタブで更新してプロシージャを表示した上で、users_info を実行すると、GUI ウィンドウが表示されますので、インプットだけ入力して、実行します。すると指定した ID のユーザーの email がアウトプットとして表示されました。

このようにアウトプットは表示する内容を決定することができます。

■ストアドファンクションの活用

こちらのコマンドを確認していきます。

```
DELIMITER $$
CREATE FUNCTION zei_sum(input INT) RETURNS FLOAT(10,2)
DETERMINISTIC
BEGIN
      DECLARE zei FLOAT(3,2);
    SET zei = 1.10;
    RETURN input * zei;
END$$
DELIMITER ;
```

と入力して実行します。このクエリは、DECLARE で 2 桁の小数点まで表示するデータ型を設定して、1.10 を消費税値として zei に設定することで、入力したインプット値の税込み価格を計算してくれる関数となります。

次にこのファンクションを利用します。ストアドプロシージャとは異なり CALL ではなく SELECT を利用します。

```
SELECT zei_sum(1300);
```

と入力して、税込み計算を行います。
このように税込み価格を実行してくれました。この関数は通常のクエリの中で利用することができます。

```
SELECT zei_sum(price) FROM products;
```
と入力して、製品価格を税込み価格でデータ抽出します。このように SUM や COUNT などの既存の関数と同じように利用できることがわかります。

この zei_sum 関数の実行内容にはクエリを利用していませんが、関数の設定にクエリを利用する方法を学習します。

```
DELIMITER $$
CREATE FUNCTION zei_sum(input INT) RETURNS FLOAT(10,2)
DETERMINISTIC
BEGIN
  DECLARE zei FLOAT(3,2);
   SET zei = 1.10;

   SELECT price FROM products
   WHERE id = input;

   RETURN price * zei;
END$$
DELIMITER ;
```

と入力して、プロダクト ID をインプットにして、その製品の価格の消費税を計算する関数を作成します。

これを実行するとエラーとなってしまいます。Not allowed to return a result set from a function エラーと表示されているかと思いますが、これは SELECT から抽出されるデータを直接に RETURN 内で利用できないために発生するエラーです。

そのため、これを修正して、エラーが発生し奈ようにするためには、INTO 句を利用して変数に代入することが必要です。

```
DELIMITER $$
CREATE FUNCTION zei_sum(input INT) RETURNS FLOAT(10,2)
DETERMINISTIC
BEGIN
   DECLARE p_price INT(5);
    DECLARE zei FLOAT(3,2);
   SET zei = 1.10;

   SELECT price INTO p_price FROM products
   WHERE id = input;
```

```
    RETURN p_price * zei;
END$$
DELIMITER ;
```

DECLARE p_price INT(5);という変数を追加して、ここに SELECT 文で抽出される価格データを代入するようにします。SELECT price INTO p_price FROM products として、p_price に代入します。

RETURN では p_price * zei;という計算値に変更します。これで実行すると、実行が上手くいきます。

SELECT zei_sum(1);
でインプットを設定して実行すると、プロダクト ID 1 の価格が消費税込みで表示されました。このようにクエリを駆使しして、ユーザーが定義する関数を作ることができます。