Advanced SQL

Team 2

Chong Shen, Peifeng Li, Jiajun Chen, Xiangnan Zhang, Kebin Li, Fan Shen

This homework is done in MySQL

1. Add a new column called "price" to "products" table

```
ALTER TABLE Products ADD price INT;
```

2.Add a new column called "price" to "order_products" table

```
ALTER TABLE Order_products ADD price INT;
```

3.Using SQL Cursor fill "price" column in "products" table with a random number between 1 to 1000

```
DELIMITER //

CREATE PROCEDURE add_product_prices()
BEGIN
DECLARE seq INT;
DECLARE done INT DEFAULT TRUE;
DECLARE cursor_results CURSOR FOR SELECT product_id FROM Products FOR UPDATE;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = FALSE;
OPEN cursor_results;
       FETCH cursor_results INTO seq;
       WHILE done
              DO BEGIN
                     UPDATE Products SET price=FLOOR(1+rand()*1000) WHERE
product_id=seq;
             FETCH cursor_results INTO seq;
              END;
       END WHILE;
CLOSE cursor_results;
END//

CALL add_product_prices();
```

product_id	product_name	aisle_id	price
1	Chocolate Sandwich Cookies	61	721
2	All-Seasons Salt	104	780
3	Robust Golden Unsweetened Oolong Tea	94	524
4	Smart Ones Classic Favorites Mini Rigatoni Wit...	38	610
5	Green Chile Anytime Sauce	5	413
6	Dry Nose Oil	11	529
7	Pure Coconut Water With Orange	98	770
8	Cut Russet Potatoes Steam N' Mash	116	137
9	Light Strawberry Blueberry Yogurt	120	501
10	Sparkling Orange Juice & Prickly Pear Beverage	115	296
11	Peach Mango Juice	31	320
12	Chocolate Fudge Layer Cake	119	398
13	Saline Nasal Mist	11	183
14	Fresh Scent Dishwasher Cleaner	74	170
15	Overnight Diapers Size 6	56	878
16	Mint Chocolate Flavored Syrup	103	353
17	Rendered Duck Fat	35	864
18	Pizza for One Suprema Frozen Pizza	79	267
19	Gluten Free Quinoa Three Cheese & Mushroom...	63	897
20	Pomegranate Cranberry & Aloe Vera Enrich Drink	98	526
21	Small & Medium Dental Dog Treats	40	917

4.Using SQL script fill "price" column in "order_products" table with the corresponding price in "products" table

```
DELIMITER //

CREATE PROCEDURE add_orderproduct_prices()
```

```
BEGIN
DECLARE seq INT;
DECLARE done INT DEFAULT TRUE;
DECLARE cursor_results CURSOR FOR SELECT product_id FROM Order_products FOR UPDATE;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = FALSE;
OPEN cursor_results;
       FETCH cursor_results INTO seq;
       WHILE done
              DO BEGIN
                     UPDATE Order_products SET price = (SELECT price FROM Products
WHERE product_id=seq) WHERE product_id=seq;
                     FETCH cursor_results INTO seq;
              END;
       END WHILE;
CLOSE cursor_results;
END//

CALL add_orderproduct_prices();
```

Cross Check

```
SELECT distinct product_id, price from Order_products order by product_id;
```

product_id	price
1	721
2	780
3	524
4	610
5	413
6	529
7	770
8	137
9	501
10	296
11	320
12	398
13	183
14	170
15	878
16	353
17	864
18	267
19	897
20	526

5.Write one sample SQL script for each of the following SQL Control-Flow Statements using Instacart database

- **Begin transaction**

```
BEGIN
START TRANSACTION;
       INSERT INTO Aisles VALUES(300, 'tt', 3);
    SAVEPOINT cp1;
    DELETE FROM Aisles WHERE aisle_id = 300;
    ROLLBACK TO SAVEPOINT cp1;
COMMIT;
END;
```

124	spirits	5
125	trail mix snack mix	19
126	feminine care	11
127	body lotions soap	11
128	tortillas flat bread	3
129	frozen appetizers sides	1
130	hot cereal pancake mixes	14
131	dry pasta	9
132	beauty	11
133	muscles joints pain relief	11
134	specialty wines champagnes	5
300	ttt	3
NULL	NULL	NULL

- **Begin-end**

```
BEGIN
	SELECT * FROM Aisles;
END;
(1) If-else
BEGIN
	IF(SELECT COUNT(*) FROM Aisles) > 100
	THEN SELECT '>100';
	ELSE SELECT '<100';
	END IF;
END;
```

Result Grid

	> 100
▶	> 100

- **While-break-continue**
-

```
BEGIN
labela: WHILE (SELECT price from Products where product_id = 1)
	DO BEGIN
		UPDATE Products SET price = price * 2 WHERE product_id = 1;
			IF (SELECT price FROM Products WHERE product_id = 1) > 4000
			THEN LEAVE labela;
		ELSE ITERATE labela;
		END IF;
	END;
	END WHILE;
END;
```

	product_id	product_name	aisle_id	Price
▶	1	Chocolate Sandwich Cookies	61	7272
	2	All-Seasons Salt	104	321
	3	Robust Golden Unsweetened Oolong Tea	94	614

- **Goto**

Since MyQSL does not support Goto, so I write an example which can run on SQL server.

```
DECLARE @num INT;
BEGIN
	SET @num = (SELECT COUNT(*) FROM Aisles)
    IF @num > 100 GOTO Branch_one
    IF @num < 100 GOTO Branch_two
END

Branch_one: SELECT 'Branch_one'
Branch_two: SELECT 'Branch_two'
TheEnd: SELECT 'The_end'
```

- **Return**

```
CREATE FUNCTION 'return' RETURNS INT(11)
BEGIN
DECLARE num INT;
SET num = 100;
IF(SELECT COUNT(*) FROM Aisles) > num THEN RETURN 1;
ELSE RETURN 0;
END IF;
END;
```

	instacart.return()
▶	1

- **Case**

```
CREATE PROCEDURE case_sample()
BEGIN
SELECT product_id, product_name, price,
       CASE WHEN price = 0 THEN 'Not for resale'
       WHEN price < 50 THEN 'under 50'
       WHEN price >= 50 AND price < 500 THEN 'under500'
       WHEN price >= 500 AND price < 1000 THEN 'under 1000'
       ELSE 'Over 1000'
       END
       FROM Products;
END;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

product_id	product_name	Price	Case when Price = 0 then 'Not for resale' when Price < 50 then 'under 50' when Price >= 50 and Price < 500 then 'under 500' when Price >= 500 and Price < 1000 then 'under 1000' Else 'Over 1000'
1	Chocolate Sandwich Cookies	608	under 1000
2	All-Seasons Salt	314	under 500
3	Robust Golden Unsweetened Oolong Tea	553	under 1000
4	Smart Ones Classic Favorites Mini Rigatoni With ...	266	under 500
5	Green Chile Anytime Sauce	264	under 500
6	Dry Nose Oil	925	under 1000
7	Pure Coconut Water With Orange	445	under 500

Result 1

- **Waitfor**

```
BEGIN
       SELECT SLEEP(5);
       SELECT * FROM Products WHERE product_id = 1;
END;
```

28 12:18:59 call instacart.waitfor() 1 row(s) returned 5.032 sec / 0.000 sec

- **Try...catch**

```
BEGIN
DECLARE EXIT HANDLER FOR SQLEXEPTION
       BEGIN
              SELECT 'No column product in Products';
       END;
       SELECT product_name FROM Products;
       SELECT * FROM Aisles;
END;
```

No column product in Products
No column product in Products

- **Throw**

```
BEGIN
DECLARE EXIT HANDLER FOR SQLEXEPTION
       BEGIN
              SIGNAL SQLSTATE VALUE '99999'
              SET MESSAGE_TEST = 'An error occurred'
       END;
       SELECT product_name FROM Products
       SELECT * FROM Aisles;
```

```
END;
```
57 12:49:54 call instacart.throw() Error Code: 1644. An error occurred

6.CREATE/ALTER/DROP/ENABLE/DISABLE a DDL Trigger

Mysql does not support DDL trigger. I write it in sql server format.

- **Create**

```
CREATE TRIGGER DDL_TRIGGER ON Instacart
FOR DROP_TABLE AS
PRINT 'YOU CAN NOT DROP TABLE'
ROLLBACK;
```

- **Drop**

```
DROP TRIGGER DDL_TRIGGER ON Instacart;
```

- **Alter**

```
ALTER TRIGGER DDL_TRIGGER ON Instacart
FOR ALTER_TABLE AS
PRINT 'YOU CAN NOT ALTER TABLE'
ROLLBACK;
```

- **Enable**

```
ENABLE TRIGGER DDL_TRIGGER ON Instacart;
```

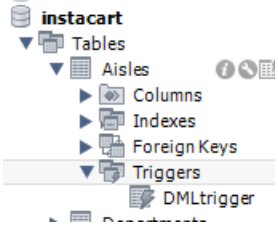
- **Disable**

```
DISABLE TRIGGER DDL_TRIGGER ON Instacart;
```

7.CREATE/ALTER/DROP/ENABLE/DISABLE a DML Trigger

- **create**

```
CREATE TRIGGER DML_TRIGGER BEFORE INSERT ON Aisles
FOR EACH ROW
SIGNAL SQLSTATE VALUE '99999'
MESSAGE_TEXT = 'An error occurred';
```



- **drop**

```
DROP TRIGGER DML_TRIGGER
```

- **Alter**

(MySQL do not support alter, enable and disable just write down how it should be like)

```
ALTER TRIGGER DML_TRIGGER BEFORE INSERT ON Aisles DML_trigger
FOR EACH ROW
       SIGNAL SQLSTATE VALUE '99999'
       SET MESSAGE_TEXT = 'not allowed to insert data';
```

- **Enable**

```
ENABLE TRIGGER DML_TRIGGER;

ENABLE TRIGGER DML_TRIGGER ON Aisles; --sql server syntax
```

- **Disable**

```
DISABLE TRIGGER DML_TRIGGER;

DISABLE TRIGGER DML_TRIGGER ON Aisles; --sql server syntax
```