付録 3 実習問題解答例

4.1.10 実習 SELECT 句

1.解答

```
> SELECT * FROM product;
```

2.解答

```
> SELECT prod_name, cost, discount FROM product;
```

3.解答

```
> SELECT prod_name, cost, discount, cost * discount AS 割引価格 FROM product;
```

4.解答

```
> SELECT prod_name, cost, discount, cost * coalesce(discount, 1) AS 割引価格
  FROM product
  ;
```

5.解答

```
> SELECT prod_name, cost, discount,
  format((cost * case WHEN discount IS NULL THEN 1 ELSE discount END), 2)
  AS 割引価格 FROM product;
```

6.解答①

```
> SELECT prod_name, cost, discount,
  format((cost * case WHEN discount IS NULL THEN 1 ELSE discount END), 2)
  AS 割引価格 FROM product ORDER BY 2 DESC;
```

6.解答②

```
> SELECT prod_name, cost, discount,
  format((cost * case WHEN discount IS NULL THEN 1 ELSE discount END), 2)
  AS 割引価格 FROM product ORDER BY cost DESC;
```

7.解答

```
> SELECT prod_name, cost, discount,
  format((cost * case WHEN discount IS NULL THEN 1 ELSE discount END), 2)
  AS 割引価格 FROM product ORDER BY 2 DESC LIMIT 3;
```

4.2.7　実習 WHERE 句

1.解答

```
> SELECT * FROM product WHERE cost >= 20000;
```

2.解答

```
> SELECT cust_id, cust_name FROM customer WHERE fax IS NULL;
```

3.解答

```
> SELECT sales_no, psales_no, prod_id, price FROM sales
  WHERE psales_no BETWEEN 110 AND 119 ORDER BY price DESC;
```

4.解答①

```
> SELECT prod_id, prod_name, cost*discount FROM product
  WHERE prod_id IN (102, 104, 106)
  ORDER BY cost*discount DESC;
```

4.解答②

```
> SELECT prod_id, prod_name, cost*discount FROM product
  WHERE prod_id IN (102, 104, 106)
  ORDER BY 3 DESC;
```

5.解答①

```
> SELECT cust_id, cust_address, delivery_date FROM packedsales
  WHERE cust_address LIKE '%渋谷区%'
  AND delivery_date BETWEEN '1996-01-01' AND '1996-12-31';
```

5.解答②

```
> SELECT cust_id, cust_address, delivery_date FROM packedsales
  WHERE cust_address LIKE '%渋谷区%'
  AND EXTRACT(YEAR FROM delivery_date) = 1996;
```

6.解答①

```
> SELECT cust_id, cust_name, tel FROM customer
  WHERE NOT (tel LIKE '03%' OR tel LIKE '06%');
```

6.解答②

```
> SELECT cust_id, cust_name, tel FROM customer
  WHERE tel NOT LIKE '03%' AND tel NOT LIKE '06%';
```

4.3.4　実習 列関数/グループ

1.解答

```
> SELECT sum(cost), avg(cost), min(cost), max(cost) FROM product;
```

2.解答

```
> SELECT count(*), count(discount), count(DISTINCT discount),
         sum(discount), avg(discount), min(discount), max(discount)
  FROM product;
```

3.解答

```
> SELECT discount, count(discount), sum(discount), avg(discount), min(discount),
         max(discount)
  FROM product GROUP BY discount ORDER BY discount;
```

4.解答

```
> SELECT discount, count(discount), sum(discount), avg(discount), min(discount),
         max(discount)
  FROM product GROUP BY discount HAVING count(discount) >= 5 ORDER BY discount;
```

5.解答

```
> SELECT discount, count(discount), sum(discount), avg(discount), min(discount),
         max(discount)
  FROM product WHERE cost >= 15000 GROUP BY discount ORDER BY discount;
```

6.解答

```
> SELECT emp_id, avg(total) FROM packedsales GROUP BY emp_id ORDER BY emp_id;
```

7.解答

```
> SELECT psales_date, sum(total) FROM packedsales
  GROUP BY psales_date ORDER BY 2 DESC LIMIT 5;
```

8.解答

```
> SELECT DISTINCT emp_id FROM packedsales;
```

4.4.5　実習 結合

1.解答①

```
> SELECT cust_name, address, delivery_date
  FROM packedsales ps JOIN customer c ON ps.cust_id = c.cust_id
  WHERE ps.psales_no = 3;
```

1.解答②

```
> SELECT cust_name, address, delivery_date
  FROM packedsales ps JOIN customer c USING(cust_id)
  WHERE ps.psales_no = 3;
```

1.解答③

```
> SELECT cust_name, address, delivery_date
  FROM packedsales ps NATURAL JOIN customer c WHERE ps.psales_no = 3;
```

2.解答①

```
> SELECT prod_name, price * quantity AS amount
  FROM sales s JOIN product p ON s.prod_id = p.prod_id
  WHERE quantity >= 3;
```

2.解答②

```
> SELECT prod_name, price * quantity AS amount
  FROM sales s JOIN product p USING(prod_id)
  WHERE quantity >= 3;
```

2.解答③

```
> SELECT prod_name, price * quantity AS amount
  FROM sales s    NATURAL JOIN product p WHERE quantity >= 3;
```

3.解答①

```
> SELECT p.psales_no, p.psales_date, p.total, s.prod_id, s.quantity, s.price
  FROM packedsales p JOIN sales s ON p.psales_no = s.psales_no
                     JOIN customer c ON p.cust_id = c.cust_id
  WHERE c．cust_name LIKE ’田中%’；
```

3.解答②

```
> SELECT p.psales_no, p.psales_date, p.total, s.prod_id, s.quantity, s.price
  FROM packedsales p JOIN sales s USING(psales_no)
                     JOIN customer c USING(cust_id)
  WHERE c.cust_name LIKE '田中%';
```

3.解答③

```
> SELECT p.psales_no, p.psales_date, p.total, s.prod_id, s.quantity, s.price
  FROM packedsales p NATURAL JOIN sales s
                     NATURAL JOIN customer c
  WHERE c.cust_name LIKE '田中%';
```

4.解答

```
> SELECT e.emp_id, e.emp_name, avg(ps.total)
  FROM packedsales ps JOIN employee e ON ps.emp_id = e.emp_id
  GROUP BY e.emp_id, e.emp_name ORDER BY e.emp_id;
```

5.解答

```
> SELECT p.prod_id, p.prod_name, sum(s.quantity), sum(s.price)
  FROM sales s JOIN product p ON s.prod_id = p.prod_id
  GROUP BY p.prod_id, p.prod_name HAVING sum(s.price) >= 500000
  ORDER BY p.prod_id;
```

6.解答①

```
> SELECT prod_name, sum(price * quantity) AS total
  FROM sales s JOIN product p ON s.prod_id = p.prod_id
  GROUP BY prod_name ORDER BY total DESC;
```

6.解答②

```
> SELECT prod_name, sum(price * quantity) AS total
  FROM sales s JOIN product p USING(prod_id)
  GROUP BY prod_name ORDER BY total DESC;
```

6.解答③

```
> SELECT prod_name, sum(price * quantity) AS total
  FROM sales s NATURAL JOIN product p GROUP BY prod_name ORDER BY total DESC;
```

7.解答①

```
> SELECT cust_name, cust_address, delivery_date, prod_name, quantity
  FROM packedsales ps  JOIN customer c  ON ps.cust_id = c.cust_id
                       JOIN sales s     ON ps.psales_no = s.psales_no
                       JOIN product p   ON s.prod_id = p.prod_id
  WHERE ps.psales_no = 3;
```

7.解答②

```
> SELECT cust_name, cust_address, delivery_date, prod_name, quantity
  FROM packedsales ps  JOIN customer c  USING(cust_id)
                       JOIN sales s     USING(psales_no)
                       JOIN product p   USING(prod_id)
  WHERE ps.psales_no = 3;
```

7.解答③

```
> SELECT cust_name, cust_address, delivery_date, prod_name, quantity
  FROM packedsales ps  NATURAL JOIN customer c
                       NATURAL JOIN sales s
                       NATURAL JOIN product p
  WHERE ps.psales_no = 3;
```

8.解答

```
> SELECT address FROM customer UNION SELECT loc FROM department;
```

9.解答

```
> SELECT *, cost * coalesce(discount, 1) * 0.95 FROM product WHERE prod_id % 2 = 0
  UNION
  SELECT *, cost * coalesce(discount, 1)        FROM product WHERE prod_id % 2 = 1
  ORDER BY 6 DESC;
```

4.5.6　実習 副照会

1.解答

```
> SELECT psales_no, prod_id, price FROM sales
  WHERE price = (SELECT max(price) FROM sales);
```

2.解答

```
> SELECT psales_no, emp_id, cust_id, total FROM packedsales
  WHERE total > (SELECT avg(total) FROM packedsales)
  ORDER BY total, psales_no;
```

3.解答

```
> SELECT sales_no, prod_id, price FROM sales
  WHERE sales_no = 1 AND price <= ALL (SELECT avg(price) FROM sales
  GROUP BY prod_id);
```

4.解答①

```
> SELECT dept_id, dept_name FROM department d
  WHERE NOT EXISTS (SELECT * FROM employee WHERE dept_id = d.dept_id);
```

4.解答②

```
> SELECT dept_id, dept_name FROM department
  WHERE dept_id NOT IN (SELECT dept_id FROM employee);
```

5.1.1　実習 INSERT

1.解答

```
> INSERT INTO employee VALUES
  (100, 10, '丸野 一夫', '1972-07-01', CURRENT_DATE, 1, 5000, NULL);
```

2.解答

```
> INSERT INTO customer(cust_id, cust_name)
  SELECT emp_id + 1000, emp_name FROM employee;
```

5.2.1　実習 UPDATE

1.解答

```
> UPDATE department SET loc = '神奈川県川崎市' WHERE dept_id BETWEEN 20 AND 30;
```

2.解答

```
> UPDATE department, (SELECT emp_id FROM employee
    WHERE sal = (SELECT min(sal) FROM employee)) as tmp
    SET department.mgr_id = tmp.emp_id WHERE adept_id = 40;
```

5.3.1　実習 DELETE

1.解答

```
> DELETE FROM product WHERE discount is NULL;
```

2.解答

```
> DELETE FROM product;
```

6.2　実習 TRUNCATE

1.解答

```
> TRUNCATE department;
```

7.2.4 実習 トランザクション

1.解答

Q.振込みの動作の間の 10 秒間に、ターミナルを強制終了してみてください。ふたたびターミナルを立ち上げたときデータはどうなっているでしょうか?

A.つぎのように、最初の UPDATE 文だけが反映されます。

```
> SELECT * FROM account;
 emp_id | balance
--------+---------
      6 |   900
      8 |  1000
2 rows in set (0.00 sec)
```

2.解答

Q.トランザクションが異常終了したときに、データが元に戻るようにするには、どうしたらよいでしょうか。

A.このスクリプトをトランザクションとして実行します。すなわち、スクリプト実行前に START TRANSACTION 文を実行します。

```
> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

> SOURCE TransfarAccountData.sql
```

ふたたびターミナルを立ち上げて、account 表を確認すると、ロールバックしています。データは、このトランザクションを行っていない状態に戻ります。

```
> SELECT * FROM account;
 emp_id | balance
--------+---------
      6 |  1000
      8 |  1000
2 rows in set (0.00 sec)
```

なお、実際には、このスクリプトの後に、COMMIT 文を入れておきます。

8.1.4　実習 データベース

1.解答

```
> CREATE DATABASE restaurant;
```

2.解答

```
> DROP DATABASE restaurant;
```

10.1.4　実習 CREATETABLE

1.解答

```
> DELETE FROM employee WHERE dept_id =20;
> DELETE FROM department WHERE dept_id = 20;
```

10.1.6　実習 ALTER TABLE

1.解答

```
> ALTER TABLE packedsales ADD FOREIGN KEY(cust_id) REFERENCES customer(cust_id);
> SHOW COLUMNS FROM packedsales;
 Field         | Type         | Null | Key | Default | Extra
---------------+--------------+------+-----+---------+-------
 psales_no     | int(11)      | NO   | PRI | NULL    |
 psales_date   | date         | YES  |     | NULL    |
 emp_id        | int(11)      | YES  |     | NULL    |
 cust_id       | int(11)      | YES  |     | NULL    |
 cust_address  | varchar(40)  | YES  |     | NULL    |
 cust_tel      | varchar(20)  | YES  |     | NULL    |
 delivery_date | date         | YES  |     | NULL    |
 delivery_time | time         | YES  |     | NULL    |
 total         | decimal(9,2) | YES  |     | NULL    |
 carriage      | decimal(9,2) | YES  |     | NULL    |
 excise        | decimal(9,2) | YES  |     | NULL    |
```

2.解答

```
> INSERT INTO packedsales(psales_no, cust_id) VALUES (100, 10);
Query OK, 1 row affected (0.02 sec)
```

3.解答

```
> UPDATE packedsales SET cust_id = 100 WHERE psales_no = 1;
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint f
ails (`sample`.`packedsales`, CONSTRAINT `packedsales_ibfk_1` FOREIGN KEY (`cust
_id`) REFERENCES `customer` (`cust_id`))
```

4.解答

```
> UPDATE customer SET cust_id = 100 WHERE cust_id = 1;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constrai
nt fails (`sample`.`packedsales`, CONSTRAINT `packedsales_ibfk_1` FOREIGN KEY (`
cust_id`) REFERENCES `customer` (`cust_id`))
```

5.解答

```
> UPDATE customer SET cust_id = 100 WHERE cust_id = 11;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

6.解答

```
> DELETE FROM customer WHERE cust_id = 1;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constrai
nt fails (`sample`.`packedsales`, CONSTRAINT `packedsales_ibfk_1` FOREIGN KEY (`
cust_id`) REFERENCES `customer` (`cust_id`))
```

10.1.8 実習 DROP TABLE

1.解答

```
> DROP TABLE packedsales CASCADE;
```

11.1.2　実習 CREATE VIEW

1.解答

```
> CREATE VIEW saleslist AS
  SELECT DISTINCT e.emp_id, e.emp_name, p.prod_id, p.prod_name, c.cust_id,
                  c.cust_name
  FROM employee e JOIN packedsales ps ON e.emp_id = ps.emp_id
                  JOIN customer c ON ps.cust_id = c.cust_id
                  JOIN sales s    ON ps.psales_no = s.psales_no
                  JOIN product p  ON s.prod_id = p.prod_id;
```

11.1.4　実習 DROP VIEW

1.解答

```
> DROP VIEW saleslist [CASCADE];
```

12.1.2　実習 CREATE INDEX

1.解答

```
> CREATE INDEX cust_name_index ON customer(cust_name);
```

12.1.4　実習 DROP INDEX

1.解答

```
> DROP INDEX cust_name_index ON customer;
```

13.1.5　実習 ストアドプログラム

1.解答

```
> DELIMITER //
> DROP FUNCTION IF EXISTS ADD_NUM //
> CREATE FUNCTION ADD_NUM(param1 INT, param2 INT) RETURNS INT
  BEGIN
  RETURN (param1 + param2);
  END
> //
> DELIMITER ;
```

2.解答

```
> DELIMITER //
> DROP FUNCTION IF EXISTS SELECT_SAL //
> CREATE FUNCTION SELECT_SAL(param INT) RETURNS INT
  BEGIN
  DECLARE result INT;
  SELECT sal INTO result FROM employee WHERE emp_id = param;
  RETURN result;
  END
> //
> DELIMITER ;
```

3.解答

```
> DELIMITER //
> DROP FUNCTION IF EXISTS SELECT_SAL2 //
> CREATE FUNCTION SELECT_SAL2(param INT) RETURNS VARCHAR(50)
  BEGIN
  DECLARE result VARCHAR(50);
  DECLARE name   VARCHAR(20);
  DECLARE s INT;
  SELECT emp_name, sal INTO name, s FROM employee WHERE emp_id = param;
  SET result = CONCAT(name, 'さんの給料は', CAST(s AS CHAR), '円です。');
  RETURN result;
  END
> //
> DELIMITER ;
```

4.解答

```
> DELIMITER //
> DROP PROCEDURE IF EXISTS ADD_NUM //
> CREATE PROCEDURE ADD_NUM(IN param1 INT, param2 INT)
  BEGIN
  SELECT (param1 + param2) AS 'SAL';
  END
> //
> DELIMITER ;
```

5.解答

```
> DELIMITER //
> DROP PROCEDURE IF EXISTS SELECT_EMPNAME //
> CREATE PROCEDURE SELECT_EMPNAME(IN id INT)
  BEGIN
  DECLARE name   VARCHAR(20);
  DECLARE EXIT HANDLER FOR SQLSTATE '42S02' SELECT '結果がありません' AS 'error';
  SELECT emp_name INTO name FROM employee WHERE emp_id = id;
  IF name IS NOT NULL THEN
  SELECT name AS name;
    ELSE
      SELECT * FROM AAA;
    END IF;
  END
> //
> DELIMITER ;
```

13.2.4　実習 制御文

1.解答

```
> DELIMITER //
> DROP PROCEDURE IF EXISTS EMPLIST2 //
> CREATE PROCEDURE EMPLIST2()
  BEGIN
  DECLARE name   VARCHAR(20);
  DECLARE count  INT DEFAULT 1;
  DECLARE emplist2 CURSOR FOR SELECT emp_name FROM employee;
  OPEN emplist2;
    fetch_loop: LOOP
    FETCH emplist2 INTO name;
      IF name LIKE '工藤 新一' THEN
         SELECT name AS '検索結果';
      END IF;
      IF count = 0 THEN
         LEAVE fetch_loop;
      END IF;
    END LOOP;
    CLOSE emplist2;
  END
> //
> DELIMITER ;
```