1. usr テーブルからすべてのデータを表示する
SELECT * FROM usr;
2. usr テーブルから「family」が 4 以上の「uname」「family」を表示する
SELECT uname, family
FROM usr
WHERE family >= 4;
3. usr テーブルから「family」に NULL が含まれる「uname」「family」を表示する
SELECT uname, family
FROM usr
WHERE family IS NULL;
4. usr テーブルから「uid」に to が含まれる「uname」「family」を表示する
SELECT uname, family
FROM usr
WHERE uid LIKE '%to%';

5. usr テーブルから「family」が 4 より小さく、
「uid」が da で終わる「uid」「uname」「family」を表示する
SELECT uid, uname, family
FROM usr
WHERE family < 4
  AND uid LIKE '%da';
6. usr テーブルから「family」が 5 より大きいデータを削除する
DELETE FROM usr
WHERE family > 5;

7. usr テーブルの「family」に NULL を含むデータの「family」を 0 に変更する
UPDATE usr
SET family = 0
WHERE family IS NULL;

8. usr テーブルに「uid」が test、
「passwd」が this、
「uname」が pattern になるデータを追加する
INSERT INTO usr (uid, passwd, uname)
VALUES ('test', 'this', 'pattern');

9. usr テーブルから、「family」が 3 の
「uname」「family」を「uid」の小さい順に表示する

```
SELECT uname, family
FROM usr
WHERE family = 3
ORDER BY uid ASC;
```

10. usr テーブルから、
「family」が大きい順で 3 つ、すべてのデータを表示する
```
SELECT *
FROM usr
ORDER BY family DESC
LIMIT 3;
```

11. usr テーブルの「family」の合計値、平均値を表示する
```
SELECT SUM(family), AVG(family)
FROM usr;
```

12. schedule テーブルのデータの数を
「uid」ごとのグループに分けて表示する
```
SELECT uid, COUNT(*)
FROM schedule
GROUP BY uid;
```

13. usr テーブルの「family」の平均値を
四捨五入して表示する
```
SELECT ROUND(AVG(family))
FROM usr;
```

14. schedule テーブルに「uid」が yyamada、
「subject」が test、
「pdate」が SQL を実行した日付のデータを入力する
```
INSERT INTO schedule (uid, subject, pdate)
VALUES ('yyamada', 'test', CURRENT_DATE);
```

15. usr テーブルから「uid」と
「uname」の頭から 2 文字を表示する
```
SELECT SUBSTRING(uid,1,2), SUBSTRING(uname,1,2)
FROM usr;
```

16. usr テーブルと schedule テーブルを
互いの「uid」で外部結合して、
usr の「uname」、schedule の「subject」、「pdate」を表示する。
「uname」がすべて表示されるようにする。

```
SELECT u.uname, s.subject, s.pdate
FROM usr u
LEFT JOIN schedule s
ON u.uid = s.uid;
```

17. schedule テーブルの「pid」「subject」と
category テーブルの「cname」を、
「cid」で内部結合して表示する

```
SELECT s.pid, s.subject, c.cname
FROM schedule s
INNER JOIN category c
ON s.cid = c.cid;
```

18. usr テーブルから schedule テーブルの
「cid」の平均値より低い「family」の
すべてのデータを表示する

```
SELECT *
FROM usr
WHERE family < (
    SELECT AVG(cid)
    FROM schedule
);
```

19. schedule テーブルから usr テーブルの「uid」と同じ
「uid」が登録されているデータの
「pid」「subject」を表示する

```
SELECT s.pid, s.subject
FROM schedule s
INNER JOIN usr u
ON s.uid = u.uid;
```

20. 問題なく SQL が実行されたので、
トランザクションを終了してデータを保存する

```
COMMIT;
```