解答 C.1 SQL ドリル

C.1.1 銀行口座データベース

1. SELECT 口座番号 , 名義 , 種別 , 残高 , 更新日 FROM 口座
2. SELECT 口座番号 FROM 口座
3. SELECT 口座番号 , 残高 FROM 口座
4. SELECT * FROM 口座
5. UPDATE 口座 SET 名義 = ' X X X X X '
6. UPDATE 口座 SET 残高 = 99999999, 更新日 = '2022-03-01'
7. INSERT INTO 口座 (口座番号 , 名義 , 種別 , 残高 , 更新日) VALUES ('0642191', ' アオキ　ハルカ ', '1', 3640551, '2022-03-13'); INSERT INTO 口座 (口座番号 , 名義 , 種別 , 残高 , 更新日) VALUES ('1039410', ' キノシタ　リュウジ ', '1', 259017, '2021-11-30'); INSERT INTO 口座 (口座番号 , 名義 , 種別 , 残高) VALUES ('1239855', ' タカシナ　ミツル ', '2', 6509773);
8. DELETE FROM 口座
9. SELECT * FROM 口座 WHERE 口座番号 = '0037651'
10. SELECT * FROM 口座 WHERE 残高 > 0
11. SELECT * FROM 口座 WHERE 口座番号 < '1000000'
12. SELECT * FROM 口座 WHERE 更新日 < '2022-01-01'
13. SELECT * FROM 口座 WHERE 残高 >= 1000000
14. SELECT * FROM 口座 WHERE 種別 <> '1'
15. SELECT * FROM 口座 WHERE 更新日 IS NULL
16. SELECT * FROM 口座 WHERE 名義 LIKE '% ハシ %'
17. SELECT * FROM 口座 WHERE 更新日 BETWEEN '2022-01-01' AND '2022-01-31'
18. SELECT * FROM 口座 WHERE 種別 IN ('2', '3')
19. SELECT * FROM 口座 WHERE 名義 IN (' サカタ　リョウヘイ ', ' マツモト　ミワコ ', ' ハマダ　サトシ ')
20. SELECT * FROM 口座 WHERE 更新日 >= '2021-12-30' AND 更新日 < '2022-01-05'
21. SELECT * FROM 口座 WHERE 残高 < 10000 AND 更新日 IS NOT NULL
22. SELECT * FROM 口座 WHERE 口座番号 LIKE '2_____' OR 名義 LIKE 'エ__□%コ'
23. 口座テーブルの主キーは「口座番号」、取引テーブルの主キーは「取引番号」、取引事由テーブルの主キーは「取引事由 ID」
24. SELECT 口座番号 , 名義 , 種別 , 残高 , 更新日 FROM 口座 ORDER BY 口座番号

※「22.」の「□」(白四角)は、全角空きを表しています。

25. SELECT DISTINCT 名義 FROM 口座 ORDER BY 名義
26. SELECT 口座番号 , 名義 , 種別 , 残高 , 更新日 FROM 口座 ORDER BY 4 DESC, 1
27. SELECT 更新日 FROM 口座 WHERE 更新日 IS NOT NULL ORDER BY 更新日 OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY
28. SELECT 更新日 , 残高 FROM 口座 WHERE 残高 > 0 AND 更新日 IS NOT NULL ORDER BY 残高 , 更新日 DESC OFFSET 10 ROWS FETCH NEXT 10 ROWS ONLY
29. SELECT 口座番号 FROM 口座 UNION SELECT 口座番号 FROM 廃止口座 ORDER BY 1
30. SELECT 名義 FROM 口座 EXCEPT SELECT 名義 FROM 廃止口座 ORDER BY 1 DESC
31. SELECT 名義 FROM 口座 INTERSECT SELECT 名義 FROM 廃止口座 ORDER BY 1
32. SELECT 口座番号 , 残高 FROM 口座 WHERE 残高 = 0 UNION SELECT 口座番号 , 解約時残高 FROM 廃止口座 WHERE 解約時残高 <> 0 ORDER BY 1
33. SELECT 口座番号 , 名義 , ' ○ ' AS 口座区分 FROM 口座 UNION SELECT 口座番号 , 名義 , ' × ' AS 口座区分 FROM 廃止口座 ORDER BY 名義
34. SELECT 口座番号 , 残高 / 1000 AS 千円単位の残高 FROM 口座 WHERE 残高 >= 1000000
35. INSERT INTO 口座 (口座番号 , 名義 , 種別 , 残高 , 更新日) VALUES ('0652281', ' タカギ　ノブオ ', '1', 100000 + 3000, '2022-04-01'); INSERT INTO 口座 (口座番号, 名義, 種別, 残高 , 更新日) VALUES ('1026413', ' マツモト　サワコ ', '1', 300000 + 3000, '2022-04-02'); INSERT INTO 口座 (口座番号 , 名義 , 種別 , 残高 , 更新日) VALUES ('2239710', ' ササキ　シゲノリ ', '1', 1000000 + 3000, '2022-04-03');
36. UPDATE 口座 SET 残高 = (残高 - 3000) * 1.003 WHERE 口座番号 IN ('0652281 ', '1026413', '2239710')
37. SELECT 口座番号 , 更新日 , 更新日 + 180 AS 通帳期限日 FROM 口座 WHERE 更新日 < '2021-01-01'
38. SELECT 口座番号 , ' カ)' || 名義 AS 名義 FROM 口座 WHERE 種別 = '3'
39. SELECT DISTINCT 種別 AS 種別コード , CASE 種別 WHEN '1' THEN ' 普通 ' WHEN '2' THEN ' 当座 ' WHEN '3' THEN ' 別段 ' END AS 種別名 FROM 口座
40. SELECT 口座番号 , 名義 , CASE WHEN 残高 < 100000 THEN 'C' WHEN 残高 >= 100000 AND 残高 < 1000000 THEN 'B' ELSE 'A' END AS 残高ランク FROM 口座
41. SELECT LENGTH(口座番号), LENGTH(REPLACE(名義 , '　', '')), LENGTH(CAST(残高 AS VARCHAR)) FROM 口座
42. SELECT * FROM 口座 WHERE SUBSTRING(名義 , 1, 5) LIKE '% カワ %'
43. SELECT * FROM 口座 WHERE LENGTH(CAST(残高 AS VARCHAR)) >= 4 AND SUBSTRING (CAST(残高 AS VARCHAR), LENGTH(CAST(残高 AS VARCHAR))-2, 3) = '000'
44. SELECT 口座番号 , 残高 , TRUNC(残高 * 0.0002, 0) AS 利息 FROM 口座 ORDER BY 残高 DESC
45. SELECT 口座番号 , 残高 , CASE WHEN 残高 < 500000 THEN TRUNC(残高 * 0.0001, 0) WHEN 残高 >= 500000 AND 残高 < 2000000 THEN TRUNC(残高 * 0.0002, 0) WHEN

残高 >= 2000000 THEN TRUNC(残高 * 0.0003, 0) END AS 残高別利息 FROM 口座 ORDER BY 残高別利息 DESC, 口座番号

46. INSERT INTO 口座 (口座番号 , 名義 , 種別 , 残高 , 更新日) VALUES ('0351262', ' イトカワ　ダイ ', '2', 635110, CURRENT_DATE); INSERT INTO 口座 (口座番号 , 名義 , 種別 , 残高 , 更新日) VALUES ('1015513', ' アキツ　ジュンジ ', '1', 88463, CURRENT_DATE); INSERT INTO 口座 (口座番号 , 名義 , 種別 , 残高 , 更新日) VALUES ('1739298', ' ホシノ　サトミ ', '1', 704610, CURRENT_DATE);
47. SELECT 口座番号 , 名義 , 種別 , 残高 , SUBSTRING(CAST(更新日 AS VARCHAR), 1, 4) || ' 年 ' || SUBSTRING(CAST(更新日 AS VARCHAR), 6, 2) || ' 月 ' || SUBSTRING(CAST(更新日 AS VARCHAR), 9, 2) || ' 日 ' AS 更新日 FROM 口座 WHERE 更新日 >= '2022-01-01'
48. SELECT COALESCE(CAST(更新日 AS VARCHAR), ' 設定なし ') AS 更新日 FROM 口座
49. SELECT SUM(残高) AS 合計 , MAX(残高) AS 最大 , MIN(残高) AS 最小 , AVG(残高) AS 平均 , COUNT(*) AS 件数 FROM 口座
50. SELECT COUNT(*) AS 件数 FROM 口座 WHERE 種別 <> '1' AND 残高 >= 1000000 AND 更新日 < '2022-01-01'
51. SELECT COUNT(*) - COUNT(更新日) AS 更新日が登録されていない件数 FROM 口座
52. SELECT MAX(名義), MIN(名義) FROM 口座
53. SELECT MAX(更新日), MIN(更新日) FROM 口座
54. SELECT 種別 , SUM(残高) AS 合計 , MAX(残高) AS 最大 , MIN(残高) AS 最小 , AVG(残高) AS 平均 , COUNT(*) AS 件数 FROM 口座 GROUP BY 種別
55. SELECT SUBSTRING(口座番号 , 7, 1) AS 口座番号グループ , COUNT(*) AS 件数 FROM 口座 GROUP BY SUBSTRING(口座番号 , 7, 1) ORDER BY 件数 DESC
56. SELECT SUBSTRING(COALESCE(CAST(更新日 AS VARCHAR), 'XXXX'), 1, 4) AS 更新年 , SUM(残高) AS 合計 , MAX(残高) AS 最大 , MIN(残高) AS 最小 , AVG(残高) AS 平均 , COUNT(*) AS 件数 FROM 口座 GROUP BY SUBSTRING(COALESCE(CAST(更新日 AS VARCHAR), 'XXXX'), 1, 4)
57. SELECT 種別 , SUM(残高) AS 合計 , COUNT(*) AS 件数 FROM 口座 GROUP BY 種別 HAVING SUM(残高) > 3000000
58. SELECT SUBSTRING(名義 , 1, 1) AS 名義 , COUNT(名義) AS 件数 , AVG(LENGTH (REPLACE(名義 , '　', ''))) AS 文字数の平均 FROM 口座 GROUP BY SUBSTRING(名義 , 1, 1) HAVING COUNT(名義) >= 10 OR AVG(LENGTH(REPLACE(名義 , '　', ''))) > 5
59. UPDATE 口座 SET 残高 = 残高 + (SELECT COALESCE(SUM(入金額), 0) - COALESCE (SUM(出金額), 0) FROM 取引 WHERE 口座番号 = '0351333' AND 日付 = '2022-01-11'), 更新日 = '2022-01-11' WHERE 口座番号 = '0351333'
60. SELECT 残高 , (SELECT SUM(入金額) FROM 取引 WHERE 口座番号 = '1115600' AND 日付 = '2021-12-28') AS 入金額合計 , (SELECT SUM(出金額) FROM 取引 WHERE 口座番号 = '1115600' AND 日付 = '2021-12-28') AS 出金額合計 FROM 口座 WHERE 口座番号 = '1115600'

61. SELECT 口座番号 , 名義 , 残高 FROM 口座 WHERE 口座番号 IN (SELECT 口座番号 FROM 取引 WHERE 入金額 >= 1000000)
62. SELECT * FROM 口座 WHERE 更新日 >ALL (SELECT 日付 FROM 取引)
63. SELECT A. 日付 , (SELECT MAX(入金額) FROM 取引 WHERE 口座番号 = '3104451') AS 最大入金額 , (SELECT MAX(出金額) FROM 取引 WHERE 口座番号 = '3104451') AS 最大出金額 FROM (SELECT 日付 FROM 取引 WHERE 口座番号 = '3104451' GROUP BY 日付 HAVING SUM(入金額) > 0 AND SUM(出金額) > 0) AS A
64. INSERT INTO 廃止口座 SELECT * FROM 口座 WHERE 口座番号 = '2761055'; DELETE FROM 口座 WHERE 口座番号 = '2761055';
65. SELECT T. 口座番号 , T. 日付 , J. 取引事由名 , COALESCE(T. 入金額 , T. 出金額) AS 取引金額 FROM 取引 AS T JOIN 取引事由 AS J ON T.取引事由ID = J.取引事由ID WHERE T.口座番号 IN ('0311240', '1234161', '2750902') ORDER BY T. 口座番号 , T. 取引番号
66. SELECT K. 口座番号 , K. 名義 , K. 残高 , T. 日付 , T. 入金額 , T. 出金額 FROM 口座 AS K JOIN 取引 AS T ON K. 口座番号 = T. 口座番号 WHERE K. 口座番号 = '0887132' ORDER BY T. 取引番号
67. SELECT T. 口座番号 , K. 名義 , K. 残高 FROM 取引 AS T JOIN 口座 AS K ON T. 口座番号 = K. 口座番号 WHERE T. 日付 = '2020-03-01'
68. SELECT T. 口座番号 , COALESCE(K. 名義 , ' 解約済み ') AS 名義 , COALESCE(K. 残高 , 0) AS 残高 FROM 取引 AS T LEFT JOIN 口座 AS K ON T.口座番号 = K.口座番号 WHERE T.日付 = '2020-03-01'
69. SELECT T.取引番号, CAST(J.取引事由ID AS VARCHAR) || ':' || J.取引事由名 AS 取引事由, T. 日付 , T. 口座番号 , T. 入金額 , T. 出金額 FROM 取引 AS T RIGHT JOIN 取引事由 AS J ON T. 取引事由 ID = J. 取引事由 ID
70. SELECT DISTINCT COALESCE(T. 取引事由 ID, J. 取引事由 ID), J. 取引事由名 FROM 取引 AS T FULL JOIN 取引事由 J ON T. 取引事由 ID = J. 取引事由 ID
    /* FULL JOIN が使えない場合、以下で代替 */ SELECT DISTINCT T. 取引事由 ID, J. 取引事由名 FROM 取引 AS T LEFT JOIN 取引事由 J ON T. 取引事由 ID = J. 取引事由 ID UNION SELECT DISTINCT J. 取引事由 ID, J. 取引事由名 FROM 取引 AS T RIGHT JOIN 取引事由 J ON T. 取引事由 ID = J. 取引事由 ID
71. SELECT K. 口座番号 , K. 名義 , K. 残高 , T. 日付 , J. 取引事由名 , T. 入金額 , T. 出金額 FROM 口座 AS K JOIN 取引 AS T ON K. 口座番号 = T. 口座番号 JOIN 取引事由 AS J ON T. 取引事由 ID = J. 取引事由 ID WHERE K. 口座番号 = '0887132' ORDER BY T. 取引番号
72. SELECT K. 口座番号 , K. 名義 , K. 残高 , T. 日付 , T. 取引事由 ID, T. 入金額 , T. 出金額 FROM 口座 AS K JOIN 取引 AS T ON K. 口座番号 = T. 口座番号 WHERE K. 残高 >= 5000000 AND (T. 入金額 >= 1000000 OR T. 出金額 >= 1000000) AND T. 日付 >= '2022-01-01'
73. /* 口座テーブルを副問い合わせにした場合 */ SELECT K. 口座番号 , K. 名義 , K. 残高 , T. 日付 , T. 取引事由ID, T. 入金額, T. 出金額 FROM 取引 AS T JOIN (SELECT 口座番号, 名義,

残高 FROM 口座 WHERE 残高 >= 5000000) AS K ON T. 口座番号 = K. 口座番号 WHERE (T. 入金額 >= 1000000 OR T. 出金額 >= 1000000) AND T. 日付 >= '2022-01-01'
/* 取引テーブルを副問い合わせにした場合 */ SELECT K. 口座番号 , K. 名義 , K. 残高 , T. 日付 , T. 取引事由ID, T. 入金額 , T. 出金額 FROM 口座 AS K JOIN (SELECT 口座番号 , 日付 , 取引事由 ID, 入金額 , 出金額 FROM 取引 WHERE (入金額 >= 1000000 OR 出金額 >= 1000000) AND 日付 >= '2022-01-01') AS T ON K. 口座番号 = T. 口座番号 WHERE K. 残高 >= 5000000

74. SELECT K. 口座番号 , T. 回数 , K. 名義 FROM 口座 AS K JOIN (SELECT 口座番号 , COUNT(*) AS 回数 FROM 取引 GROUP BY 口座番号 , 日付 HAVING COUNT(*) >= 3) AS T ON K. 口座番号 = T. 口座番号
75. /* 自己結合を用いた場合 */ SELECT DISTINCT K1. 名義 , K1. 口座番号 , K1. 種別 , K1. 残高 , K1. 更新日 FROM 口座 AS K1 JOIN 口座 AS K2 ON K1. 名義 = K2. 名義 WHERE K1. 口座番号 <> K2. 口座番号 ORDER BY K1. 名義 , K1. 口座番号
/* 集計関数と結合を用いた場合 */ SELECT K1. 名義 , K1. 口座番号 , K1. 種別 , K1. 残高 , K1. 更新日 FROM 口座 AS K1 JOIN (SELECT 名義 , COUNT(名義) AS 口座数 FROM 口座 GROUP BY 名義 HAVING COUNT(名義) > 1) AS K2 ON K1. 名義 = K2. 名義 ORDER BY K1. 名義 , K1. 口座番号

C.1.2 商店データベース

1. SELECT 商品コード , 商品名 , 単価 , 商品区分 , 関連商品コード FROM 商品
2. SELECT 商品名 FROM 商品
3. SELECT * FROM 注文
4. SELECT 注文番号 , 注文枝番 , 商品コード FROM 注文
5. INSERT INTO 商品 (商品コード , 商品名 , 単価 , 商品区分) VALUES ('W0461', ' 冬のあったかコート ', 12800, '1'); INSERT INTO 商品 (商品コード , 商品名 , 単価 , 商品区分) VALUES ('S0331', ' 春のさわやかコート ', 6800, '1'); INSERT INTO 商品 (商品コード , 商品名 , 単価 , 商品区分) VALUES ('A0582', ' 秋のシックなコート ', 9800, '1');
6. SELECT * FROM 商品 WHERE 商品コード = 'W1252'
7. UPDATE 商品 SET 単価 = 500 WHERE 商品コード = 'S0023'
8. SELECT * FROM 商品 WHERE 単価 <= 1000
9. SELECT * FROM 商品 WHERE 単価 >= 50000
10. SELECT * FROM 注文 WHERE 注文日 >= '2022-01-01'
11. SELECT * FROM 注文 WHERE 注文日 < '2021-12-01'
12. SELECT * FROM 商品 WHERE 商品区分 <> '1'
13. SELECT * FROM 注文 WHERE クーポン割引料 IS NULL
14. DELETE FROM 商品 WHERE 商品コード LIKE 'N%'

15. SELECT 商品コード , 商品名 , 単価 FROM 商品 WHERE 商品名 LIKE '% コート %'
16. SELECT 商品コード , 商品区分 FROM 商品 WHERE 商品区分 IN ('2', '3', '9')
17. SELECT * FROM 商品 WHERE 商品コード BETWEEN 'A0100' AND 'A0500'
18. SELECT * FROM 注文 WHERE 商品コード IN ('N0501', 'N1021', 'N0223')
19. SELECT * FROM 商品 WHERE 商品区分 = '3' AND 商品名 LIKE '% 水玉 %'
20. SELECT * FROM 商品 WHERE 商品名 LIKE '% 軽い %' OR 商品名 LIKE '% ゆるふわ %'
21. SELECT * FROM 商品 WHERE (商品区分 = '1' AND 単価 <= 3000) OR (商品区分 = '3' AND 単価 >= 10000)
22. SELECT 商品コード FROM 注文 WHERE 注文日 >= '2022-03-01' AND 注文日 < '2022-04-01' AND 数量 >= 3
23. SELECT * FROM 注文 WHERE 数量 >= 10 OR クーポン割引料 IS NOT NULL
24. 商品テーブルの主キーは「商品コード」、注文テーブルの主キーは「注文日・注文番号・注文枝番」
25. SELECT 商品コード , 商品名 FROM 商品 WHERE 商品区分 = '1' ORDER BY 商品コード DESC
26. SELECT 注文日 , 注文番号 , 注文枝番 , 商品コード , 数量 FROM 注文 WHERE 注文日 >= '2022-03-01' ORDER BY 注文日 , 注文番号 , 注文枝番
27. SELECT DISTINCT 商品コード FROM 注文 ORDER BY 商品コード
28. SELECT 注文日 FROM 注文 ORDER BY 注文日 DESC OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY
29. SELECT * FROM 商品 ORDER BY 単価 , 商品区分 , 商品コード OFFSET 5 ROWS FETCH NEXT 15 ROWS ONLY
30. SELECT * FROM 廃番商品 WHERE 廃番日 >= '2020-12-01' AND 廃番日 < '2021-01-01' UNION SELECT * FROM 廃番商品 WHERE 売上個数 > 100 ORDER BY 6 DESC
31. SELECT 商品コード FROM 商品 EXCEPT SELECT 商品コード FROM 注文 ORDER BY 1
32. SELECT 商品コード FROM 商品 INTERSECT SELECT 商品コード FROM 注文 ORDER BY 1 DESC
33. SELECT 商品コード , 商品名 , 単価 FROM 商品 WHERE 商品区分 = '9' AND 単価 <= 1000 UNION SELECT 商品コード , 商品名 , 単価 FROM 商品 WHERE 商品区分 = '9' AND 単価 > 10000 ORDER BY 3, 1
34. SELECT 商品コード , 単価 , 単価 * 0.95 AS キャンペーン価格 FROM 商品 WHERE 商品区分 = '9' ORDER BY 商品コード
35. UPDATE 注文 SET クーポン割引料 = クーポン割引料 + 300 WHERE 注文日 >= '2022-03-12' AND 注文日 < '2022-03-15' AND 数量 >= 2 AND クーポン割引料 IS NOT NULL
36. UPDATE 注文 SET 数量 = 数量 - 1 WHERE 注文番号 = '202202250126' AND 商品コード = 'W0156'
37. SELECT 注文番号 || '-' || CAST(注文枝番 AS VARCHAR) FROM 注文 WHERE 注文番号 >= '202110010001' AND 注文番号 <= '202110319999'

38. SELECT DISTINCT 商品区分 AS 区分 , CASE 商品区分 WHEN '1' THEN ' 衣類 ' WHEN '2' THEN ' 靴 ' WHEN '3' THEN ' 雑貨 ' WHEN '9' THEN ' 未分類 ' END AS 区分名 FROM 商品
39. SELECT 商品コード , 商品名 , 単価 , CASE WHEN 単価 < 3000 THEN 'S' WHEN 単価 >= 3000 AND 単価 < 10000 THEN 'M' ELSE 'L' END AS 販売価格ランク , 商品区分 || ':' || CASE 商品区分 WHEN '1' THEN ' 衣類 ' WHEN '2' THEN ' 靴 ' WHEN '3' THEN ' 雑貨 ' WHEN '9' THEN ' 未分類 ' END AS 商品区分 FROM 商品 ORDER BY 単価 , 商品コード
40. SELECT 商品名 , LENGTH(商品名) AS 文字数 FROM 商品 WHERE LENGTH(商品名) > 10 ORDER BY LENGTH(商品名)
41. SELECT 注文日 , SUBSTRING(注文番号 , 9, 4) AS 注文番号 FROM 注文
42. UPDATE 商品 SET 商品コード = 'E' || SUBSTRING(商品コード , 2, 4) WHERE SUBSTRING(商品コード , 1, 1) = 'M'
43. SELECT SUBSTRING(注文番号 , 9, 4) AS 注文番号 FROM 注文 WHERE SUBSTRING(注文番号 , 9, 4) >= '1000' AND SUBSTRING(注文番号 , 9, 4) <= '2000' ORDER BY SUBSTRING(注文番号 , 9, 4)
44. UPDATE 廃番商品 SET 廃番日 = CURRENT_DATE WHERE 商品コード = 'S1990'
45. SELECT 商品コード , 商品名 , 単価 , TRUNC(単価 * 0.7, 0) AS 値下げした単価 FROM 商品 WHERE 単価 >= 10000
46. SELECT SUM(数量) AS 数量合計 FROM 注文
47. SELECT 注文日 , SUM(数量) AS 数量合計 FROM 注文 GROUP BY 注文日 ORDER BY 注文日
48. SELECT 商品区分 , MIN(単価) AS 最小額 , MAX(単価) AS 最高額 FROM 商品 GROUP BY 商品区分 ORDER BY 商品区分
49. SELECT 商品コード , SUM(数量) AS 数量合計 FROM 注文 GROUP BY 商品コード ORDER BY 商品コード
50. SELECT 商品コード , SUM(数量) AS 数量合計 FROM 注文 GROUP BY 商品コード ORDER BY 数量合計 DESC, 商品コード OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY
51. SELECT 商品コード , SUM(数量) AS 数量合計 FROM 注文 GROUP BY 商品コード HAVING SUM(数量) < 5
52. SELECT COUNT(クーポン割引料) AS 割引件数 , SUM(クーポン割引料) AS 割引額合計 FROM 注文
53. SELECT SUBSTRING(注文番号 , 1, 6) AS 年月 , COUNT(*) AS 注文件数 FROM 注文 WHERE 注文枝番 = 1 GROUP BY SUBSTRING(注文番号 , 1, 6) ORDER BY SUBSTRING(注文番号 , 1, 6) DESC
54. SELECT 商品コード FROM 注文 WHERE 商品コード LIKE 'Z%' GROUP BY 商品コード HAVING SUM(数量) >= 100
55. SELECT 商品コード , 商品名 , 単価 , (SELECT SUM(数量) FROM 注文 WHERE 商品コード = 'S0604') AS 数量 FROM 商品 WHERE 商品コード = 'S0604'

56. UPDATE 注文 SET 商品コード = (SELECT 商品コード FROM 商品 WHERE 商品区分 = '2' AND 商品名 LIKE '% ブーツ %' AND 商品名 LIKE '% 雨 %' AND 商品名 LIKE '% 安心 %') WHERE 注文日 = '2022-03-15' AND 注文番号 = '202203150014' AND 注文枝番 = 1
57. SELECT 注文日 , 商品コード FROM 注文 WHERE 商品コード IN (SELECT 商品コード FROM 商品 WHERE 商品名 LIKE '% あったか %') ORDER BY 注文日
58. SELECT 商品コード , SUM(数量) AS 数量 FROM 注文 GROUP BY 商品コード HAVING SUM(数量) >ALL (SELECT AVG(数量) FROM 注文 GROUP BY 商品コード)
59. SELECT A. 数量合計 AS 割引による販売数 , TRUNC(A. 割引料合計 / A. 数量合計 , 0) AS 平均割引額 FROM (SELECT SUM(数量) AS 数量合計 , SUM(クーポン割引料) AS 割引料合計 FROM 注文 WHERE 商品コード = 'W0746' AND クーポン割引料 IS NOT NULL) AS A
60. INSERT INTO 注文 SELECT 注文日 , 注文番号 , MAX(注文枝番) + 1, 'S1003', 1, NULL FROM 注文 WHERE 注文日 = '2022-03-21' AND 注文番号 = '202203210080' GROUP BY 注文日 , 注文番号 ; INSERT INTO 注文 SELECT 注文日 , 注文番号 , MAX(注文枝番) + 1, 'A0052', 2, 500 FROM 注文 WHERE 注文日 = '2022-03-22' AND 注文番号 = '202203220901' GROUP BY 注文日 , 注文番号
61. SELECT T. 注文番号 , T. 注文枝番 , T. 商品コード , S. 商品名 , T. 数量 FROM 注文 AS T JOIN 商品 AS S ON T. 商品コード = S. 商品コード WHERE T. 注文番号 = '202201130115' ORDER BY T. 注文番号 , T. 注文枝番
62. SELECT T. 注文日 , T. 注文番号 , T. 注文枝番 , T. 数量 , H. 単価 * T. 数量 AS 注文金額 FROM 注文 AS T JOIN 廃番商品 AS H ON T. 商品コード = H. 商品コード WHERE T. 商品コード = 'A0009' AND T. 注文日 > H. 廃番日
63. SELECT S. 商品コード , S. 商品名 , S. 単価 , T. 注文日 , T. 注文番号 , T. 数量 , S. 単価 * T. 数量 AS 売上金額 FROM 商品 AS S JOIN 注文 AS T ON S. 商品コード = T. 商品コード WHERE S. 商品コード = 'S0604' ORDER BY T. 注文番号
64. SELECT T. 商品コード , S. 商品名 FROM 注文 AS T JOIN 商品 AS S ON T. 商品コード = S. 商品コード WHERE T. 注文日 > ='2020-08-01' AND T. 注文日 < '2020-09-01'
65. SELECT T. 商品コード , COALESCE(S. 商品名 , ' 廃番 ') AS 商品名 FROM 注文 AS T LEFT JOIN 商品 AS S ON T. 商品コード = S. 商品コード WHERE T. 注文日 > ='2020-08-01' AND T. 注文日 < '2020-09-01'
66. SELECT T. 注文日 , S. 商品コード || ':' || S. 商品名 AS 商品 , COALESCE(T. 数量 , 0) AS 数量 FROM 注文 AS T RIGHT JOIN 商品 AS S ON T. 商品コード = S. 商品コード WHERE S. 商品区分 = '3'
67. SELECT T. 注文日 , S. 商品コード || ':' || S. 商品名 AS 商品 , COALESCE(T. 数量 , 0) AS 数量 FROM 注文 AS T RIGHT JOIN (SELECT 商品コード , 商品名 , 商品区分 FROM 商品 UNION SELECT 商品コード , '(廃番済み)' AS 商品名 , 商品区分 FROM 廃番商品) AS S ON T. 商品コード = S. 商品コード WHERE S. 商品区分 = '3'
68. SELECT T. 注文日 , T. 注文番号 , T. 注文枝番 , T. 商品コード , COALESCE(S. 商品名 , H. 商

品名) AS 商品名 , COALESCE(S. 単価 , H. 単価) AS 単価 , T. 数量 AS 数量 , COALESCE(S. 単価 , H. 単価) * T. 数量 - COALESCE(T. クーポン割引料 , 0) AS 注文金額 FROM 注文 AS T LEFT JOIN 商品 AS S ON T. 商品コード = S. 商品コード LEFT JOIN 廃番商品 AS H ON T. 商品コード = H. 商品コード WHERE T. 注文番号 = '202104030010'

69. SELECT S. 商品コード , S. 商品名 , S. 単価 , COALESCE(T. 数量 , 0) AS 売上数量 , S. 単価 * COALESCE(T. 数量 , 0) AS 総売上金額 FROM 商品 AS S LEFT JOIN (SELECT 商品コード , SUM(数量) AS 数量 FROM 注文 WHERE 商品コード LIKE 'B%' GROUP BY 商品コード) AS T ON S. 商品コード = T. 商品コード WHERE S. 商品コード LIKE 'B%' ORDER BY S. 商品コード
70. SELECT S1. 商品コード , S1. 商品名 , S2. 商品コード AS 関連商品コード , S2. 商品名 AS 関連商品名 FROM 商品 AS S1 JOIN 商品 AS S2 ON S1. 商品コード = S2. 関連商品コード

C.1.3 RPG データベース

1. SELECT ID, 名称 , 職業コード , HP, MP, 状態コード FROM パーティー
2. SELECT 名称 AS なまえ , HP AS 現在の HP, MP AS 現在の MP FROM パーティー
3. SELECT * FROM イベント
4. SELECT イベント番号 AS 番号 , イベント名称 AS 場面 FROM イベント
5. INSERT INTO パーティー (ID, 名称 , 職業コード , HP, MP, 状態コード) VALUES ('A01', ' スガワラ ', '21', 131, 232, '03'); INSERT INTO パーティー (ID, 名称 , 職業コード , HP, MP, 状態コード) VALUES ('A02', ' オーエ ', '10', 156, 84, '00'); INSERT INTO パーティー (ID, 名称 , 職業コード , HP, MP, 状態コード) VALUES ('A03', ' イズミ ', '20', 84, 190, '00');
6. SELECT * FROM パーティー WHERE ID = 'C02'
7. UPDATE パーティー SET HP = 120 WHERE ID = 'A01'
8. SELECT ID, 名称 , HP FROM パーティー WHERE HP < 100
9. SELECT ID, 名称 , MP FROM パーティー WHERE MP >= 100
10. SELECT イベント番号 , イベント名称 , タイプ FROM イベント WHERE タイプ <> '3'
11. SELECT イベント番号 , イベント名称 FROM イベント WHERE イベント番号 <= 5
12. SELECT イベント番号 , イベント名称 FROM イベント WHERE イベント番号 > 20
13. SELECT イベント番号 , イベント名称 FROM イベント WHERE 前提イベント番号 IS NULL
14. SELECT イベント番号 , イベント名称 , 後続イベント番号 FROM イベント WHERE 後続イベント番号 IS NOT NULL
15. UPDATE パーティー SET 状態コード = '01' WHERE 名称 LIKE '% ミ %'
16. SELECT ID, 名称 , HP FROM パーティー WHERE HP BETWEEN 120 AND 160
17. SELECT 名称 , 職業コード FROM パーティー WHERE 職業コード IN ('01', '10', '11')

18. SELECT 名称 , 状態コード FROM パーティー WHERE 状態コード NOT IN ('00', '09')
19. SELECT * FROM パーティー WHERE HP > 100 AND MP > 100
20. SELECT * FROM パーティー WHERE ID LIKE 'A%' AND 職業コード LIKE '2%'
21. SELECT * FROM イベント WHERE タイプ ='1' AND 前提イベント番号 IS NOT NULL AND 後続イベント番号 IS NOT NULL
22. パーティーテーブルの主キーは ID、イベントテーブルの主キーはイベント番号
23. SELECT DISTINCT 状態コード FROM パーティー
24. SELECT ID, 名称 FROM パーティー ORDER BY ID
25. SELECT 名称 , 職業コード FROM パーティー ORDER BY 名称 DESC
26. SELECT 名称 , HP, 状態コード FROM パーティー ORDER BY 状態コード , HP DESC
27. SELECT タイプ , イベント番号 , イベント名称 , 前提イベント番号 , 後続イベント番号 FROM イベント ORDER BY 1, 2
28. SELECT * FROM パーティー ORDER BY HP DESC OFFSET 0 ROWS FETCH NEXT 3 ROWS ONLY
29. SELECT * FROM パーティー ORDER BY MP DESC OFFSET 2 ROWS FETCH NEXT 1 ROWS ONLY
30. SELECT イベント番号 FROM イベント EXCEPT SELECT イベント番号 FROM 経験イベント ORDER BY 1
31. SELECT イベント番号 FROM 経験イベント WHERE クリア区分 ='1' INTERSECT SELECT イベント番号 FROM イベント WHERE タイプ ='2'
32. SELECT CASE WHEN 職業コード LIKE '1%' THEN 'S' WHEN 職業コード LIKE '2%' THEN 'M' ELSE 'A' END AS 職業区分 , 職業コード , ID, 名称 FROM パーティー ORDER BY 職業コード
33. SELECT 名称 AS なまえ , HP AS 現在の HP, HP + 50 AS 装備後の HP FROM パーティー WHERE 職業コード IN ('11', '21')
34. UPDATE パーティー SET MP = MP + 20 WHERE ID IN ('A01', 'A03')
35. SELECT 名称 AS なまえ , HP AS 現在の HP, HP * 2 AS 予想されるダメージ FROM パーティー WHERE 職業コード = '11'
36. SELECT 名称 AS なまえ , HP || ' ／ ' || MP AS HP と MP, CASE 状態コード WHEN '00' THEN NULL WHEN '01' THEN ' 眠り ' WHEN '02' THEN ' 毒 ' WHEN '03' THEN ' 沈黙 ' WHEN '04' THEN '混乱' WHEN '09' THEN '気絶' END AS ステータス FROM パーティー
37. SELECT イベント番号 , イベント名称 , CASE タイプ WHEN '1' THEN ' 強制 ' WHEN '2' THEN ' フリー ' WHEN '3' THEN ' 特殊 ' END AS タイプ , CASE WHEN イベント番号 >= 1 AND イベント番号 <=10 THEN ' 序盤 ' WHEN イベント番号 >= 11 AND イベント番号 <=17 THEN ' 中盤 ' ELSE ' 終盤 ' END AS 発生時期 FROM イベント
38. SELECT 名称 AS なまえ , HP AS 現在の HP, LENGTH(名称) * 10 AS 予想ダメージ FROM パーティー
39. /* % 演算子を使った場合 */ UPDATE パーティー SET 状態コード = '04' WHERE HP % 4

= 0 OR MP % 4 = 0
/* MOD 関数を使った場合 */ UPDATE パーティー SET 状態コード = '04' WHERE MOD(HP, 4) = 0 OR MOD(MP, 4) = 0

40. SELECT TRUNC(777 * 0.7, 0) AS 支払った金額
41. UPDATE パーティー SET HP = ROUND(HP * 1.3, 0), MP = ROUND(MP * 1.3, 0)
42. SELECT 名称 AS なまえ , HP, POWER(HP, 0) AS 攻撃 1 回目 , POWER(HP, 1) AS 攻撃 2 回目 , POWER(HP, 2) AS 攻撃 3 回目 FROM パーティー WHERE 職業コード = '10'
43. SELECT 名称 AS なまえ , HP,　状態コード , CASE WHEN HP <= 50 THEN 3 + CAST(状態コード AS INTEGER) WHEN HP >= 51 AND HP <= 100 THEN 2 + CAST(状態コード AS INTEGER) WHEN HP >= 101 AND HP <= 150 THEN 1 + CAST(状態コード AS INTEGER) ELSE CAST(状態コード AS INTEGER) END AS リスク値 FROM パーティー ORDER BY リスク値 DESC, HP
44. SELECT COALESCE(CAST(前提イベント番号 AS VARCHAR), ' 前提なし ') AS 前提イベント番号 , イベント番号 , COALESCE(CAST(後続イベント番号 AS VARCHAR), ' 後続なし ') AS 後続イベント番号 FROM イベント ORDER BY イベント番号
45. SELECT MAX(HP) AS 最大 HP, MIN(HP) AS 最小 HP, AVG(HP) AS 平均 HP, MAX(MP) AS 最大 MP, MIN(MP) AS 最小 MP, AVG(MP) AS 平均 MP FROM パーティー
46. SELECT CASE タイプ WHEN '1' THEN ' 強制 ' WHEN '2' THEN ' フリー ' WHEN '3' THEN ' 特殊 ' END AS タイプ , COUNT(イベント番号) AS イベント数 FROM イベント GROUP BY タイプ
47. SELECT クリア結果 , COUNT(イベント番号) AS イベント数 FROM 経験イベント WHERE クリア区分 ='1' GROUP BY クリア結果 ORDER BY クリア結果
48. SELECT CASE WHEN SUM(MP) < 500 THEN ' 敵は見とれている！ ' WHEN SUM(MP) >= 500 AND SUM(MP) < 1000 THEN ' 敵は呆然としている！ ' WHEN SUM(MP) >= 1000 THEN ' 敵はひれ伏している！ ' END AS 小さな奇跡 FROM パーティー
49. SELECT CASE クリア区分 WHEN '1' THEN ' クリアした ' WHEN '0' THEN ' 参加したがクリアしていない ' END AS 区分 , COUNT(イベント番号) AS イベント数 FROM 経験イベント GROUP BY クリア区分
50. SELECT SUBSTRING(職業コード , 1, 1) AS 職業タイプ , MAX(HP) AS 最大 HP, MIN(HP) AS 最小 HP, AVG(HP) AS 平均 HP, MAX(MP) AS 最大 MP, MIN(MP) AS 最小 MP, AVG(MP) AS 平均 MP FROM パーティー GROUP BY SUBSTRING(職業コード , 1, 1)
51. SELECT SUBSTRING(ID, 1, 1) AS ID による分類 , AVG(HP) AS HP の平均 , AVG(MP) AS MP の平均 FROM パーティー GROUP BY SUBSTRING(ID, 1, 1) HAVING AVG(HP) > 100
52. SELECT SUM(CASE WHEN HP < 100 THEN 1 WHEN HP >= 100 AND HP < 150 THEN 2 WHEN HP >= 150 AND HP < 200 THEN 3 WHEN HP >= 200 THEN 5 END) AS 開けられる扉の数 FROM パーティー
53. SELECT 名称 AS なまえ , HP AS 現在の HP, ROUND(CAST(HP AS NUMERIC) / (SELECT SUM(HP) FROM パーティー) * 100, 1) AS パーティーでの割合 FROM パーティー

WHERE 職業コード = '01'

54. UPDATE パーティー SET MP = MP + (SELECT ROUND(SUM(MP * 0.1)) FROM パーティー WHERE 職業コード <> '20') WHERE 職業コード = '20'
55. SELECT イベント番号 , クリア結果 FROM 経験イベント WHERE クリア区分 = '1' AND イベント番号 IN (SELECT イベント番号 FROM イベント WHERE タイプ IN ('1', '3'))
56. SELECT 名称 , MP FROM パーティー WHERE MP = (SELECT MAX(MP) FROM パーティー)
57. SELECT イベント番号 , イベント名称 FROM イベント WHERE イベント番号 NOT IN (SELECT イベント番号 FROM 経験イベント) ORDER BY イベント番号
58. SELECT COUNT(*) AS 未着手イベントの数 FROM (SELECT イベント番号 FROM イベント EXCEPT SELECT イベント番号 FROM 経験イベント) AS SUB
59. SELECT イベント番号 , イベント名称 FROM イベント WHERE イベント番号 < (SELECT イベント番号 FROM 経験イベント WHERE ルート番号 = 5)
60. SELECT イベント番号 , イベント名称 , 前提イベント番号 FROM イベント WHERE 前提イベント番号 =ANY (SELECT イベント番号 FROM 経験イベント WHERE クリア区分 = '1')
61. /* イベント番号 9 の更新 */ UPDATE 経験イベント SET クリア区分 = '1', クリア結果 = 'B' ルート番号 = (SELECT MAX(ルート番号) + 1 FROM 経験イベント) HERE イベント番号 = 9 /* 後続イベントの登録 */ INSERT INTO 経験イベント VALUES ((SELECT イベント番号 FROM イベント WHERE 前提イベント番号 = 9), '0', NULL, NULL)
62. SELECT E. ルート番号 , E. イベント番号 , M. イベント名称 , E. クリア結果 FROM 経験イベント E JOIN イベント M ON E. イベント番号 = M. イベント番号 WHERE クリア区分 = '1' ORDER BY E. ルート番号
63. SELECT M. イベント番号 , M. イベント名称 , E. クリア区分 FROM イベント M JOIN 経験イベント E ON M. イベント番号 = E. イベント番号 WHERE タイプ ='1'
64. SELECT M. イベント番号 , M. イベント名称 , COALESCE(E. クリア区分 , ' 未クリア ') AS クリア区分 FROM イベント M LEFT JOIN 経験イベント E ON M. イベント番号 = E. イベント番号 WHERE タイプ ='1'
65. SELECT P.ID, P. 名称 AS なまえ , S. コード名称 AS 職業 , J. コード名称 AS 状態 FROM パーティー P JOIN (SELECT コード値 , コード名称 FROM コード WHERE コード種別 ='1') S ON P. 職業コード = S. コード値 JOIN (SELECT コード値 , コード名称 FROM コード WHERE コード種別 ='2') J ON P. 状態コード = J. コード値 ORDER BY ID
66. SELECT P.ID, COALESCE(P. 名称 , ' 仲間になっていない！ ') AS なまえ , S. コード名称 AS 職業 FROM パーティー P RIGHT JOIN (SELECT コード値 , コード名称 FROM コード WHERE コード種別 ='1') S ON P. 職業コード = S. コード値
67. SELECT E. イベント番号 , E. クリア区分 , C. コード値 || '：' || C. コード名称 AS クリア結果 FROM 経験イベント E FULL JOIN (SELECT コード値 , コード名称 FROM コード WHERE コード種別 ='4') C ON E. クリア結果 = C. コード値

/*FULL JOIN が使えない場合、以下で代替 */
SELECT E. イベント番号 , E. クリア区分 , E. クリア結果 || ':' || C. コード名称 AS クリア結果 FROM 経験イベント E LEFT JOIN (SELECT コード値 , コード名称 FROM コード WHERE コード種別 ='4') C ON E. クリア結果 = C. コード値 UNION SELECT E. イベント番号 , E. クリア区分 , C. コード値 || ':' || C. コード名称 AS クリア結果 FROM 経験イベント E RIGHT JOIN (SELECT コード値 , コード名称 FROM コード WHERE コード種別 ='4') C ON E. クリア結果 = C. コード値

68. SELECT E1. イベント番号 , E1. イベント名称 , E1. 前提イベント番号 , E2. イベント名称 AS 前提イベント名称 FROM イベント E1 JOIN イベント E2 ON E1. 前提イベント番号 = E2. イベント番号 WHERE E1. 前提イベント番号 IS NOT NULL

69. SELECT E1. イベント番号 , E1. イベント名称 , E1. 前提イベント番号 , E2. イベント名称 AS 前提イベント名称 , E1. 後続イベント番号 , E3. イベント名称 AS 後続イベント名称 FROM イベント E1 LEFT JOIN イベント E2 ON E1. 前提イベント番号 = E2. イベント番号 LEFT JOIN イベント E3 ON E1. 後続イベント番号 = E3. イベント番号 WHERE E1. 前提イベント番号 IS NOT NULL OR E1. 後続イベント番号 IS NOT NULL

70. SELECT E. イベント番号 , E. イベント名称 , Z. 前提イベント数 FROM イベント E JOIN (SELECT 前提イベント番号 , COUNT(前提イベント番号) AS 前提イベント数 FROM イベント WHERE 前提イベント番号 IS NOT NULL GROUP BY 前提イベント番号) Z ON E. イベント番号 = Z. 前提イベント番号 ORDER BY E. イベント番号

解答 C.2 正規化ドリル

C.2.1 基礎問題

第 2 正規形から第 3 正規形へ

1. **会員** ＝ 会員 ID ＋ 会員名 ＋ 所属ジム ID(FK)
   ジム ＝ ジム ID ＋ ジム名

2. **医療機関** ＝ 医療機関コード ＋ 医療機関名 ＋ 系列会 ID(FK)
   系列会 ＝ 系列会 ID ＋ 系列会名

3. **スマホアプリ** ＝ アプリ ID ＋ アプリ名 ＋ 紹介文 ＋ 開発者 ID(FK)
   開発者 ＝ 開発者 ID ＋ 開発者名

4. **紙幣** ＝ 紙幣番号 ＋ 種別 ID(FK)
   種別 ＝ 種別 ID ＋ 種別名 ＋ 額面

5. **機械学習モデル** ＝ モデル管理 ID ＋ モデル種別 ID(FK) ＋ 学習開始日
   ＋ 学習終了日 ＋ 学習用データセット ID(FK)
   機械学習モデル種別 ＝ モデル種別 ID ＋ モデル種別名
   学習用データセット ＝ 学習用データセット ID ＋ 学習用データセット名

第 1 正規形から第 2 正規形へ

6. **給与支払** ＝ 支払年月 ＋ 社員 ID(FK) ＋ 支払総額
   社員 ＝ 社員 ID ＋ 社員名

7. **チケット** ＝ 上映作品 ID(FK) ＋ 上映開始日時 ＋ シアター番号
   上映作品 ＝ 上映作品 ID ＋ 作品名

8. **導入 DBMS** ＝ DBMS 製品名 (FK) ＋ 導入バージョン
   DBMS 製品 ＝ DBMS 製品名 ＋ 最新バージョン ＋ 製造元

9. **フライト** ＝ 日付 ＋ 国内定期運行便名 (FK)

国内定期運航便　＝　国内定期運航便名　＋　発地空港コード　＋　着地空港コード
＋　離陸予定時刻

10. **自治体**　＝　都道府県番号 (FK)　＋　市町村番号 (FK)
都道府県　＝　都道府県番号　＋　都道府県名　＋　知事名
市町村　＝　市町村番号　＋　市町村名　＋　市町村長名

非正規形から第 1 正規形へ

11. **外来予約**　＝　予約日時　＋　担当医氏名
予約明細　＝　予約日時 (FK)　＋　診察券番号　＋　患者名

12. **宛先**　＝　宛先コード　＋　郵便番号　＋　住所
宛名　＝　宛先コード (FK)　＋　宛名　＋　敬称

13. **ダンジョン**　＝　ダンジョン ID　＋　ダンジョン名
登場モンスター　＝　ダンジョン ID(FK)　＋　登場モンスター ID
＋　登場モンスター名　＋　最大H P

14. **レシート**　＝　レジ番号　＋　レシート連番　＋　発行日
レシート明細　＝　レジ番号 (FK)　＋　レシート連番 (FK)　＋　発行日 (FK)
＋　商品番号　＋　商品名　＋　価格

15. **サブスク契約**　＝　契約番号　＋　契約者 ID　＋　契約者名　＋　契約日
プラン明細　＝　契約番号 (FK)　＋　プラン ID　＋　プラン名　＋　月額単価
オプション明細　＝　契約番号 (FK)　＋　割引オプション ID　＋　割引オプション名

16. **交換用レンズ**　＝　レンズ型番　＋　レンズ名称　＋　焦点距離　＋　F 値
レンズ対応カメラ　＝　レンズ型番 (FK)　＋　対応カメラ型番　＋　対応カメラ機種名
レンズ対応カメラ製造工場　＝　レンズ型番 (FK)　＋　カメラ型番 (FK)
＋　製造工場 ID　＋　工場名
レンズ製造工場　＝　レンズ型番 (FK)　＋　製造工場 ID　＋　工場名

※この後の第 2 正規化で、「レンズ対応カメラ製造工場」「レンズ製造工場」のそれぞれから分離される「製造工場」は、同一の存在として統合される可能性がある。

C.2.2 総合問題

以下に示す解答例は、唯一の正解ではありません。ユーザービューをどのように読み取るか、テーブル名や列名をどのように命名するか、人工キーを補うか複合主キーを用いるかなどによって、複数の妥当な設計を考えることができます。

題材（1）書籍リスト

非正規形

書籍リスト ＝ ISBN ＋ タイトル ＋ 定価 ＋ 発売日 ＋ 出版社 ID
＋ 出版社名 ＋ （著者 ID ＋ 著者名）＊

第 1 正規形

書籍リスト ＝ ISBN ＋ タイトル ＋ 定価 ＋ 発売日 ＋ 出版社 ID ＋ 出版社名
著者関連付け ＝ ISBN(FK) ＋ 著者 ID ＋ 著者名

第 2 正規形

書籍リスト ＝ ISBN ＋ タイトル ＋ 定価 ＋ 発売日 ＋ 出版社 ID ＋ 出版社名
著者関連付け ＝ ISBN(FK) ＋ 著者 ID(FK)
著者 ＝ 著者 ID ＋ 著者名

第 3 正規形

書籍リスト ＝ ISBN ＋ タイトル ＋ 定価 ＋ 発売日 ＋ 出版社 ID(FK)
著者関連付け ＝ ISBN(FK) ＋ 著者 ID(FK)
著者 ＝ 著者 ID ＋ 著者名
出版社 ＝ 出版社 ID ＋ 出版社名

題材（2）タイムライン

非正規形

タイムライン ＝ ユーザー名 ＋ ユーザー ID ＋ （投稿 ID ＋ 投稿者名
＋ 投稿者 ID ＋ 投稿時刻 ＋ 本文）＊

第 1 正規形

ユーザー ＝ ユーザー ID ＋ ユーザー名
タイムライン ＝ ユーザー ID(FK) ＋ 投稿 ID ＋ 投稿者 ID ＋ 投稿者名
＋ 投稿時刻 ＋ 本文

第 2 正規形

ユーザー　＝　ユーザー ID　＋　ユーザー名
タイムライン　＝　ユーザー ID(FK)　＋　投稿 ID(FK)
投稿　＝　投稿 ID　＋　投稿者 ID　＋　投稿者名　＋　投稿時間　＋　本文

第 3 正規形

ユーザー　＝　ユーザー ID　＋　ユーザー名
タイムライン　＝　ユーザー ID(FK)　＋　投稿 ID(FK)
投稿　＝　投稿 ID　＋　投稿者 ID(FK)　＋　投稿時間　＋　本文
投稿者　＝　投稿者 ID　＋　投稿者名

※「ユーザー」テーブルと「投稿者」テーブルは実質的に同一の存在として、以下のように統合してもよい。
ユーザー　＝　ユーザー ID　＋　ユーザー名
タイムライン　＝　ユーザー ID(FK)　＋　投稿 ID(FK)
投稿　＝　投稿 ID　＋　投稿者のユーザー ID(FK)　＋　投稿時間　＋　本文

題材（3）職務経歴書

非正規形

職務経歴書　＝　経歴書番号　＋　作成日付　＋　氏名　＋　要約　＋　（経歴番号
＋　経歴期間自　＋　経歴期間至　＋　企業 ID　＋　企業名
＋　事業内容　＋　（業務番号　＋　業務期間自　＋　業務期間至
＋　業務内容）＊）＊

※ユーザービューには「職務経歴」は 1 件しか書かれていないが、期間別に複数の職歴が列挙されることを推定できるため、繰り返し項目とした。

第 1 正規形

職務経歴書　＝　経歴書番号　＋　作成日付　＋　氏名　＋　要約
職務経歴　＝　経歴書番号 (FK)　＋　経歴番号　＋　職歴開始年月　＋　職歴終了年月
＋　企業 ID　＋　企業名　＋　事業内容
業務明細　＝　経歴書番号 (FK)　＋　経歴番号 (FK)　＋　業務番号　＋　業務開始年月
＋　業務終了年月　＋　業務内容

第 2 正規形

部分関数従属している列はないため、第 1 正規形と同じ。

第 3 正規形

職務経歴書　＝　経歴書番号　＋　作成日付　＋　氏名　＋　要約

職務経歴 ＝ 経歴書番号 (FK) ＋ 経歴番号 ＋ 職歴開始年月 ＋ 職歴終了年月
＋ 企業 ID(FK)
業務明細 ＝ 経歴書番号 (FK) ＋ 経歴番号 (FK) ＋ 業務番号 ＋ 業務開始年月
＋ 業務終了年月 ＋ 業務内容
企業 ＝ 企業 ID ＋ 企業名 ＋ 事業内容

題材（4）不動産物件

非正規形

不動産物件 ＝ 物件 ID ＋ 種別コード ＋ 種別名 ＋ 賃料 ＋ 礼金 ＋ 敷金
＋ 保証金 ＋ 物件名 ＋ 所在地 ＋ 築年数 ＋ 階数
＋ 部屋番号 ＋ 間取り ＋ （部屋タイプ ＋ 部屋タイプ名
＋ 部屋面積）＊ ＋ （路線 ID ＋ 路線名 ＋ 駅番号 ＋ 駅名
＋ 所要時間）＊ ＋ （画像 ID ＋ 画像 ＋ キャプション）＊

※情報数が多い場合、ユーザービューに配置されている位置に関わらず、できるだけ関連する情報の順に並べておくほうが、この後の正規化がしやすくなる。

第 1 正規形

不動産物件 ＝ 物件 ID ＋ 種別コード ＋ 種別名 ＋ 賃料 ＋ 礼金 ＋ 敷金
＋ 保証金 ＋ 物件名 ＋ 所在地 ＋ 築年数 ＋ 階数
＋ 部屋番号 ＋ 間取り
間取り明細 ＝ 物件 ID(FK) ＋ 部屋タイプ ＋ 部屋タイプ名 ＋ 部屋面積
交通 ＝ 物件 ID(FK) ＋ 路線 ID ＋ 駅番号 ＋ 路線名 ＋ 駅名 ＋ 所要時間
物件画像 ＝ 物件 ID(FK) ＋ 画像 ID ＋ 画像 ＋ キャプション

第 2 正規形

不動産物件 ＝ 物件 ID ＋ 種別コード ＋ 種別名 ＋ 賃料 ＋ 礼金 ＋ 敷金
＋ 保証金 ＋ 物件名 ＋ 所在地 ＋ 築年数 ＋ 階数
＋ 部屋番号 ＋ 間取り
間取り明細 ＝ 物件 ID(FK) ＋ 部屋タイプ (FK) ＋ 部屋面積
部屋 ＝ 部屋タイプ ＋ 部屋タイプ名
交通 ＝ 物件 ID(FK) ＋ 路線 ID(FK) ＋ 駅番号 (FK) ＋ 所要時間
路線 ＝ 路線 ID ＋ 路線名
路線駅 ＝ 路線 ID(FK) ＋ 駅番号 ＋ 駅名
物件画像 ＝ 物件 ID(FK) ＋ 画像 ID ＋ キャプション
画像 ＝ 画像 ID ＋ 画像

※「駅」は、通常、何らかの「路線」に属していることを前提に、「路線駅」テーブルとした。また、近隣施設などの画像を複数の物件で共通して利用することを考慮して「画像」テーブルを作成した。

第 3 正規形

不動産物件　＝　物件 ID　＋　物件種別コード (FK)　＋　賃料　＋　礼金　＋　敷金
＋　保証金　＋　物件名　＋　所在地　＋　築年数　＋　階数
＋　部屋番号　＋　間取り

種別　＝　種別コード　＋　種別名

間取り明細　＝　物件 ID(FK)　＋　部屋タイプ (FK)　＋　部屋面積

部屋　＝　部屋タイプ　＋　部屋タイプ名

交通　＝　物件 ID(FK)　＋　路線 ID(FK)　＋　駅番号 (FK)　＋　所要時間

路線　＝　路線 ID　＋　路線名

路線駅　＝　路線 ID(FK)　＋　駅番号　＋　駅名

物件画像　＝　物件 ID(FK)　＋　画像 ID　＋　キャプション

画像　＝　画像 ID　＋　画像

題材 (5) 路線図

本設問では、ユーザービューの捉え方によって複数のアプローチが考えられるため、解答例を 2 つ示す。

【解答例 1　「駅」の集合を「路線」と捉える方式】

非正規形

路線図　＝　路線図 ID　＋　路線図名　＋　(路線 ID　＋　路線名　＋　路線カラー
＋　(駅番号　＋　駅名　＋　(乗換路線 ID　＋　乗換駅番号
＋　乗換駅名)＊)＊)＊

※烏丸御池駅の例に加え、三線以上の乗換駅も表現可能とするために、各駅は「乗換可能駅」の情報を複数持つ構造とした。

第 1 正規形

路線図　＝　路線図 ID　＋　路線図名

掲載路線　＝　路線図 ID(FK)　＋　路線 ID　＋　路線名　＋　路線カラー

路線駅　＝　路線 ID(FK)　＋　駅番号　＋　駅名

乗換可能駅　＝　路線 ID(FK)　＋　駅番号 (FK)　＋　乗換路線 ID(FK)　＋　乗換駅番号 (FK)
＋　乗換駅名

第 2 正規形

路線図　＝　路線図 ID　＋　路線図名

掲載路線　＝　路線図 ID(FK)　＋　路線 ID(FK)

路線　＝　路線 ID　＋　路線名　＋　路線カラー

路線駅　＝　路線 ID(FK)　＋　駅番号　＋　駅名

乗換可能駅 ＝ 路線 ID(FK) ＋ 駅番号 (FK) ＋ 乗換先の路線 ID(FK) ＋ 乗換先の駅番号 (FK)

※「乗換可能駅」テーブルの「乗換先の路線 ID」「乗換先の駅番号」は、「路線駅」テーブルの「路線 ID」「駅番号」と実質的に同一のため、「路線駅」テーブルに統合した。

第 3 正規形

推移関数従属している列はないため、第 2 正規形と同じ。

【解答例 2　路線図を「駅」の集合と「駅間線路」の集合として捉える方式】

非正規形

路線図 ＝ 路線図 ID ＋ 路線図名
駅 ＝ 駅 ID ＋ 駅名 ＋ (路線 ID ＋ 路線名 ＋ 路線カラー ＋ 路線駅番号)＊
駅間線路 ＝ 端点 A の駅 ID ＋ 端点 B の駅 ID

※路線図を「点と、点同士を結ぶ線」の集合（無向グラフ）として捉える場合、上記のような①路線図名などの表題部、②「点」である駅、③「線」である駅間線路という独立した 3 つの情報が直接導かれる。より厳密には、駅テーブルと駅間線路テーブルには、属する路線図の路線図 ID を加えて複合主キーとする。今回は、路線とは線路の集合ではなく、駅の集合として整理した。

第 1 正規形

路線図 ＝ 路線図 ID ＋ 路線図名
駅 ＝ 駅 ID ＋ 駅名
路線駅 ＝ 駅 ID(FK) ＋ 路線 ID ＋ 路線名 ＋ 路線カラー ＋ 路線駅番号
駅間線路 ＝ 端点 A の駅 ID ＋ 端点 B の駅 ID

第 2 正規形

路線図 ＝ 路線図 ID ＋ 路線図名
駅 ＝ 駅 ID ＋ 駅名
路線駅 ＝ 駅 ID(FK) ＋ 路線 ID(FK) ＋ 路線駅番号
路線 ＝ 路線 ID ＋ 路線名 ＋ 路線カラー
駅間線路 ＝ 端点 A の駅 ID ＋ 端点 B の駅 ID

第 3 正規形

推移関数従属している列はないため、第 2 正規形と同じ。

※解答例 2 では、解答例 1 とは異なり、烏丸御池駅が物理的に 1 つの存在であることを表現できるが、「物理的に異なるが乗換対象となる複数の駅」（たとえば、東京メトロの永田町駅・赤坂見附駅、大阪メトロの梅田駅・東梅田駅・西梅田駅の関係）は表現できない。

題材（6）時刻表

非正規形

時刻表　＝　時刻表 ID　＋　路線 ID　＋　路線名　＋　駅番号　＋　駅名　＋　番線
＋　方面　＋　平日休日区分　＋　平日休日区分名　＋　（出発時刻
＋　列車種別　＋　列車種別名　＋　行先駅番号　＋　行先駅名）＊　＋　備考

※時刻表の見た目から、「時」の繰り返しの中に「分」の繰り返しがあるように思えるが、列車が出発する時間は「時」と「分」の組み合わせであることに注意。従って、非正規形は平易な一重の繰り返しとなる。

第 1 正規形

時刻表　＝　時刻表 ID　＋　路線 ID　＋　路線名　＋　駅番号　＋　駅名　＋　番線
＋　方面　＋　平日休日区分　＋　平日休日区分名　＋　備考

出発列車　＝　時刻表 ID(FK)　＋　出発時刻　＋　列車種別　＋　列車種別名
＋　行先駅番号　＋　行先駅名

第 2 正規形

部分関数従属している列はないため、第 1 正規形と同じ。

第 3 正規形

時刻表　＝　時刻表 ID　＋　路線 ID(FK)　＋　駅番号 (FK)　＋　番線　＋　方面
＋　平日休日区分 (FK)　＋　備考

路線　＝　路線 ID　＋　路線名

路線駅　＝　路線 ID(FK)　＋　駅番号　＋　駅名

平日休日　＝　平日休日区分　＋　平日休日区分名

出発列車　＝　時刻表 ID(FK)　＋　出発時刻　＋　列車種別 (FK)　＋　行先駅番号 (FK)

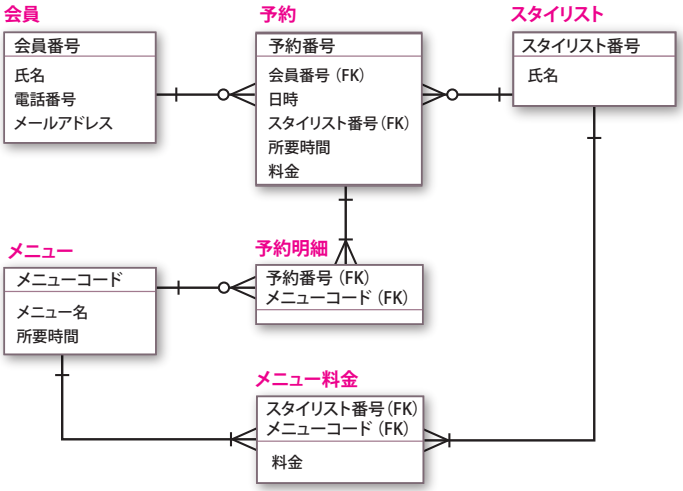
列車種別　＝　列車種別　＋　列車種別名

※「出発列車」テーブルの「行先駅番号」は、「路線駅」テーブルの「駅番号」と実質的に同じため、「路線駅」テーブルに統合した。

解答 C.3 総合問題

C.3.1 ヘアサロン予約管理データベースの作成

1．概念の整理



2. ユーザービューからのエンティティ導出

予約

予約番号
受付日時
会員番号
初回
予約日
開始時刻
所要時間
担当スタイリスト
合計金額
備考

会員

会員番号
氏名
電話番号
メールアドレス
入会日

メニュー

メニューコード
メニュー名
所要時間

スタイリスト

スタイリスト番号
氏名
入社日
ランク

予約明細

予約番号 メニュー

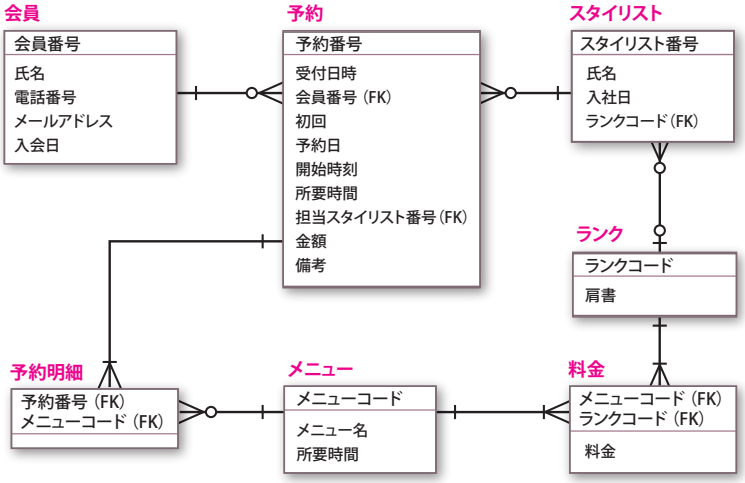
料金

メニューコード ランク
料金

ランク

ランク
肩書

3. 論理設計の完成



4. 物理設計の完成

会員の定義書

エンティティ名（論理）	会員
エンティティ名（物理）	Member

No	属性		制約			データ型		初期値	備考
	論理名	物理名	PK	FK	NN	型名	長さ		
1	会員番号	MemberNo	○		○	CHAR	4		4 桁の数字
2	氏名	MemberName			○	VARCHAR	20		
3	電話番号	Tel				CHAR	11		
4	メールアドレス	Mail				VARCHAR	40		
5	入会日	JoinDate			○	DATE		現在の日付	

ランクの定義書

エンティティ名（論理）	ランク
エンティティ名（物理）	Rank

No	属性		制約			データ型		初期値	備考
	論理名	物理名	PK	FK	NN	型名	長さ		
1	ランクコード	RankCD	○		○	CHAR	1		英字 1 文字
2	肩書	Title			○	VARCHAR	40		

スタイリストの定義書

エンティティ名（論理）	スタイリスト
エンティティ名（物理）	Stylist

No	属性		制約			データ型		初期値	備考
	論理名	物理名	PK	FK	NN	型名	長さ		
1	スタイリスト番号	StylistNo	○		○	CHAR	2		2 桁の数字
2	氏名	StylistName			○	VARCHAR	20		
3	入社日	HireDate			○	DATE			
4	ランクコード	RankCD		○		CHAR	1		見習い中はランクなし

メニューの定義書

エンティティ名（論理）	メニュー
エンティティ名（物理）	Menu

No	属性		制約			データ型		初期値	備考
	論理名	物理名	PK	FK	NN	型名	長さ		
1	メニューコード	MenuCD	○		○	CHAR	1		英字 1 文字
2	メニュー名	MenuName			○	VARCHAR	40		
3	所要時間	Duration			○	INTEGER			30 分単位

料金の定義書

エンティティ名（論理）	料金
エンティティ名（物理）	Price

No	属性		制約			データ型		初期値	備考
	論理名	物理名	PK	FK	NN	型名	長さ		
1	メニューコード	MenuCD	○	○	○	CHAR	1		英字 1 文字
2	ランクコード	RankCD	○	○	○	CHAR	1		英字 1 文字
3	料金	MenuPrice			○	INTEGER			

予約の定義書

エンティティ名（論理）	予約
エンティティ名（物理）	Reservation

No	属性		制約			データ型		初期値	備考
	論理名	物理名	PK	FK	NN	型名	長さ		
1	予約番号	ReserveNo	○		○	INTEGER			
2	受付日時	RegistDate			○	DATETIME		現在の日時	
3	会員番号	MemberNo		○	○	CHAR	4		
4	初回	First			○	BOOLEAN		0	0:既存　1:新規
5	予約日	ReserveDate			○	DATE			
6	開始時刻	StartTime			○	TIME			
7	所要時間	ServiceTime				INTEGER			30 分単位
8	担当スタイリスト番号	StylistNo		○	○	CHAR	2		
9	金額	Amount			○	INTEGER		0	
10	備考	Remarks				VARCHAR	50		

予約明細の定義書

エンティティ名（論理）	予約明細
エンティティ名（物理）	ReserveDetail

No	属性		制約			データ型		初期値	備考
	論理名	物理名	PK	FK	NN	型名	長さ		
1	予約番号	ReserveNo	○	○	○	INTEGER			
2	メニューコード	MenuCD	○	○	○	CHAR	1		英字 1 文字

解答は一例です。特に、初回の予約かどうかといった二者択一の情報を管理する属性は、通常、BIT 型のような 0 または 1 の値のみを格納できる型や、1 桁の CHAR 型で実現が可能です。使用目的や今後の機能拡張などを検討して決定します。また、長さについては、すでに業務で登録されたデータがあれば、それを参考に決めていきます。

5. DDL の作成

```
-- 会員
CREATE TABLE Member (
  MemberNo   CHAR(4)     PRIMARY KEY,
  MemberName VARCHAR(20) NOT NULL,
  Tel        CHAR(11),
  Mail       VARCHAR(40),
  JoinDate   DATE        NOT NULL DEFAULT CURRENT_DATE
);

-- ランク
CREATE TABLE Rank (
  RankCD     CHAR(1)     PRIMARY KEY,
  Title      VARCHAR(40) NOT NULL
);

-- スタイリスト
CREATE TABLE Stylist (
  StylistNo   CHAR(2)     PRIMARY KEY,
  StylistName VARCHAR(20) NOT NULL,
  HireDate    DATE        NOT NULL,
  RankCD      CHAR(1)     REFERENCES Rank(RankCD)
);

-- メニュー
CREATE TABLE Menu (
  MenuCD      CHAR(1)     PRIMARY KEY,
  MenuName    VARCHAR(40) NOT NULL,
  Duration    INTEGER     NOT NULL
);
```

```
-- 料金
CREATE TABLE Price (
  MenuCD      CHAR(1) NOT NULL REFERENCES Menu(MenuCD),
  RankCD      CHAR(1) NOT NULL REFERENCES Rank(RankCD),
  MenuPrice   INTEGER NOT NULL,
  PRIMARY KEY(MenuCD, RankCD)
);

-- 予約
CREATE TABLE Reservation (
  ReserveNo   INTEGER   PRIMARY KEY,
  RegistDate  TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  MemberNo    CHAR(4)   NOT NULL REFERENCES Member(MemberNo),
  First       BOOLEAN   NOT NULL DEFAULT '0',
  ReserveDate DATE      NOT NULL,
  StartTime   TIME      NOT NULL,
  ServiceTime INTEGER,
  StylistNo   CHAR(2)   NOT NULL REFERENCES Stylist(StylistNo),
  Amount      INTEGER   NOT NULL DEFAULT 0,
  Remarks     VARCHAR(50)
);

-- 予約明細
CREATE TABLE ReserveDetail (
  ReserveNo INTEGER NOT NULL REFERENCES Reservation(ReserveNo),
  MenuCD    CHAR(1) NOT NULL REFERENCES Menu(MenuCD),
  PRIMARY KEY(ReserveNo, MenuCD)
);
```

※外部キー制約のあるテーブルの場合、参照先のテーブルを先に作成しておく必要があります。

C.3.2 予約管理データベースの利用

6. データの登録

```
-- Member
INSERT INTO Member VALUES ('0001', '吉田康子', '0901112215',
'yoshida@a1.com', '2004-04-10');
INSERT INTO Member VALUES ('0002', '荒木和子', '0901112216',
'araki@a2.com', '2016-08-11');
INSERT INTO Member VALUES ('0003', '下田正一', '0901112217',
'shimoda@a3.com', '2017-04-12');
INSERT INTO Member VALUES ('0004', '風間由美子', '0901112218',
NULL, '2017-06-13');
INSERT INTO Member VALUES ('0005', '秋山美奈', '0901112219',
'akiyama@a5.com', '2019-01-14');
INSERT INTO Member VALUES ('0006', '木下博之', '0901112220',
'kinoshita@a6.com', '2019-04-15');
INSERT INTO Member VALUES ('0007', '広瀬正隆', NULL, NULL, '2020-
09-16');
INSERT INTO Member VALUES ('0008', '斉藤美紀', '0901112222',
'saitou@a8.co m', '2022-04-17');

-- Rank
INSERT INTO Rank VALUES ('A', 'チーフスタイリスト');
INSERT INTO Rank VALUES ('B', 'トップスタイリスト');
INSERT INTO Rank VALUES ('C', 'スタイリスト');

-- Stylist
INSERT INTO Stylist VALUES ('01', '秋葉ちか', '2002-04-01', 'A');
INSERT INTO Stylist VALUES ('02', '佐藤茜', '2004-06-01', 'B');
INSERT INTO Stylist VALUES ('03', '井上博之', '2007-01-08', 'B');
```

```
INSERT INTO Stylist VALUES ('04', '小島正', '2014-05-02', 'C');
INSERT INTO Stylist VALUES ('05', '山田雄介', '2019-04-01', 'C');
INSERT INTO Stylist VALUES ('06', '市川紀子', '2022-06-10', NULL);

-- Menu
INSERT INTO Menu VALUES ('C', 'カット', 30);
INSERT INTO Menu VALUES ('P', 'カラー', 60);
INSERT INTO Menu VALUES ('R', 'パーマ', 60);
INSERT INTO Menu VALUES ('T', 'トリートメント', 30);

-- Price
INSERT INTO Price VALUES ('C', 'A', 12000);
INSERT INTO Price VALUES ('C', 'B', 10000);
INSERT INTO Price VALUES ('C', 'C', 8000);
INSERT INTO Price VALUES ('P', 'A', 18000);
INSERT INTO Price VALUES ('P', 'B', 15000);
INSERT INTO Price VALUES ('P', 'C', 12000);
INSERT INTO Price VALUES ('R', 'A', 9600);
INSERT INTO Price VALUES ('R', 'B', 8000);
INSERT INTO Price VALUES ('R', 'C', 6400);
INSERT INTO Price VALUES ('T', 'A', 14400);
INSERT INTO Price VALUES ('T', 'B', 12000);
INSERT INTO Price VALUES ('T', 'C', 9600);

-- Reservation
INSERT INTO Reservation VALUES (1, '2022-09-06 16:28', '0002',
'0', '2022-10-01', '17:00', 90, '01', 21600, NULL);
INSERT INTO Reservation VALUES (2, '2022-09-26 12:42', '0004',
'1', '2022-10-01', '10:00', 30, '03', 10000, NULL);
INSERT INTO Reservation VALUES (3, '2022-09-30 10:30', '0008',
'0', '2022-10-01', '15:00', 150, '05', 26400, NULL);
```

```
-- ReserveDetail
INSERT INTO ReserveDetail VALUES (1, 'C');
INSERT INTO ReserveDetail VALUES (1, 'R');
INSERT INTO ReserveDetail VALUES (2, 'C');
INSERT INTO ReserveDetail VALUES (3, 'C');
INSERT INTO ReserveDetail VALUES (3, 'P');
INSERT INTO ReserveDetail VALUES (3, 'R');
```

この SQL 文により、各テーブルには次のようにデータが登録されます。

Member

MemberNo	MemberName	Tel	Mail	JoinDate
0001	吉田康子	0901112215	yoshida@a1.com	2004-04-10
0002	荒木和子	0901112216	araki@a2.com	2016-08-11
0003	下田正一	0901112217	shimoda@a3.com	2017-04-12
0004	風間由美子	0901112218	(NULL)	2017-06-13
0005	秋山美奈	0901112219	akiyama@a5.com	2019-01-14
0006	木下博之	0901112220	kinoshita@a6.com	2019-04-15
0007	広瀬正隆	(NULL)	(NULL)	2020-09-16
0008	斉藤美紀	0901112222	saitou@a8.com	2022-04-17

Rank

RankCD	Title
A	チーフスタイリスト
B	トップスタイリスト
C	スタイリスト

Stylist

StylistNo	StylistName	HireDate	RankCD
01	秋葉ちか	2002-04-01	A
02	佐藤茜	2004-06-01	B
03	井上博之	2007-01-08	B
04	小島正	2014-05-02	C
05	山田雄介	2019-04-01	C
06	市川紀子	2022-06-10	(NULL)

Menu

MenuCD	MenuName	Duration
C	カット	30
P	パーマ	60
R	カラー	60
T	トリートメント	30

Price

MenuCD	RankCD	MenuPrice
C	A	12000
C	B	10000
C	C	8000
P	A	18000
P	B	15000
P	C	12000
R	A	9600
R	B	8000
R	C	6400
T	A	14400
T	B	12000
T	C	9600

Reservation

ReserveNo	RegistDate	MemberNo	First	ReserveDate	StartTime	ServiceTime	StylistNo	Amount	Remarks
1	2022-09-06 16:28	0002	0	2022-10-01	17:00	90	01	21600	(NULL)
2	2022-09-26 12:42	0004	0	2022-10-01	12:00	30	03	10000	(NULL)
3	2022-09-30 10:30	0008	0	2022-10-01	15:00	150	05	26400	(NULL)

ReserveDetail

ReservNo	MenuCD
1	C
1	R
2	C
3	C
3	P
3	R

7. データの利用

```
1.   SELECT s.StylistName AS 名前,
            COALESCE(r.Title, 'アシスタント') AS 肩書
       FROM Stylist s
  LEFT JOIN Rank r
         ON s.RankCD = r.RankCD

2.  SELECT s.StylistName AS スタイリスト名,
           m.MenuName AS メニュー名, p.MenuPrice AS 料金
      FROM Stylist s
      JOIN Price p ON s.RankCD = p.RankCD
      JOIN Menu m  ON p.MenuCD = m.MenuCD
  ORDER BY s.RankCD, s.StylistNo, m.MenuCD

3. SELECT r.ReserveNo AS 予約番号, s.StylistName AS 担当スタイリスト名,
          m.MenuName AS メニュー名, m.Duration AS 所要時間,
          p.MenuPrice AS 料金
     FROM Reservation r
     JOIN ReserveDetail rd ON r.ReserveNo = rd.ReserveNo
     JOIN Stylist s        ON r.StylistNo = s.StylistNo
     JOIN Price p          ON s.RankCD = p.RankCD
     JOIN Menu m           ON p.MenuCD = m.MenuCD
```

```
    WHERE r.StylistNo = s.StylistNo
      AND rd.MenuCD = p.MenuCD

4.  SELECT 予約番号 , 担当スタイリスト名 , SUM( 所要時間 ) AS 合計時間 ,
           SUM( 料金 ) AS 合計金額
      FROM (SELECT r.ReserveNo AS 予約番号 ,
                   s.StylistName AS 担当スタイリスト名 ,
                   m.MenuName AS メニュー名 , m.Duration AS 所要時間 ,
                   p.MenuPrice AS 料金
              FROM Reservation r
              JOIN ReserveDetail rd ON r.ReserveNo = rd.ReserveNo
              JOIN Stylist s        ON r.StylistNo = s.StylistNo
              JOIN Price p          ON s.RankCD = p.RankCD
              JOIN Menu m           ON p.MenuCD = m.MenuCD
             WHERE r.StylistNo = s.StylistNo
               AND rd.MenuCD = p.MenuCD) t
     GROUP BY 予約番号 , 担当スタイリスト名
     ORDER BY 予約番号

5. BEGIN;
   INSERT INTO Reservation VALUES (4, '2022-10-01 10:03',
   '0006', '0', '2022-10-01', '11:30', 90, '05', 13400, NULL);
   INSERT INTO ReserveDetail VALUES (4, 'C');
   INSERT INTO ReserveDetail VALUES (4, 'R');

6. -- 所要時間の確認
   SELECT SUM(m.duration)
     FROM ReserveDetail d
     JOIN Menu m
       ON d.MenuCD = m.MenuCD
    WHERE d.ReserveNo = 4
```

```
-- 金額の確認
SELECT SUM(p.MenuPrice)
  FROM Reservation r
  JOIN ReserveDetail d
    ON r.ReserveNo = d.ReserveNo
  JOIN Stylist s
    ON r.StylistNo = s.StylistNo
  JOIN Price p
    ON d.MenuCD = p.MenuCD
   AND s.RankCD = p.RankCD
 WHERE d.ReserveNo = 4
-- トランザクション取り消し
ROLLBACK;
-- 再登録
BEGIN;
INSERT INTO Reservation VALUES (4, '2022-10-01 11:45',
'0008', '0', '2022-10-02', '11:00', 90, '05', 14400, NULL);
INSERT INTO ReserveDetail VALUES (4, 'C');
INSERT INTO ReserveDetail VALUES (4, 'R');
COMMIT;
```

7.

```
SELECT r.ReserveDate AS 予約日, s.StylistNo AS 担当スタイリスト番号,
       s.StylistName AS スタイリスト名, startTime AS 開始時刻,
       StartTime + CAST
         (ServiceTime || 'minutes' AS interval) AS 終了時刻
  FROM Reservation r
RIGHT JOIN Stylist s ON r.StylistNo = s.StylistNo
 ORDER BY r.ReserveDate, s.StylistNo
```

8.

```
SELECT r.ReserveDate AS 予約日, s.StylistNo AS 担当スタイリスト番号,
       s.StylistName AS スタイリスト名,
```

```
            EXTRACT(hour from StartTime) AS 開始時刻,
            StartTime + CAST
              (ServiceTime || 'minutes' AS interval) AS 終了時刻
        FROM Reservation r
  RIGHT JOIN Stylist s ON r.StylistNo = s.StylistNo
    ORDER BY r.ReserveDate, s.StylistNo

9.  SELECT 予約日, スタイリスト名,
          CASE WHEN 開始時刻 = 10 THEN 終了時刻 END AS "10時台",
          CASE WHEN 開始時刻 = 11 THEN 終了時刻 END AS "11時台",
          CASE WHEN 開始時刻 = 12 THEN 終了時刻 END AS "12時台",
          CASE WHEN 開始時刻 = 13 THEN 終了時刻 END AS "13時台",
          CASE WHEN 開始時刻 = 14 THEN 終了時刻 END AS "14時台",
          CASE WHEN 開始時刻 = 15 THEN 終了時刻 END AS "15時台",
          CASE WHEN 開始時刻 = 16 THEN 終了時刻 END AS "16時台",
          CASE WHEN 開始時刻 = 17 THEN 終了時刻 END AS "17時台",
          CASE WHEN 開始時刻 = 18 THEN 終了時刻 END AS "18時台"
      FROM
        (SELECT r.ReserveDate AS 予約日,
                s.StylistNo AS 担当スタイリスト番号,
                s.StylistName AS スタイリスト名,
                EXTRACT(hour from StartTime) AS 開始時刻,
                StartTime + CAST
                  (ServiceTime || 'minutes' AS interval) AS 終了時刻
           FROM Reservation r
     RIGHT JOIN Stylist s ON r.StylistNo = s.StylistNo) t
       ORDER BY CASE WHEN 予約日 IS NULL THEN 1 ELSE 0 END,
                予約日, 担当スタイリスト番号

10.  SELECT 予約日, スタイリスト名,
           MAX(CASE WHEN 開始時刻 = 10 THEN 終了時刻 END) AS "10時台",
```

```
       MAX(CASE WHEN 開始時刻 = 11 THEN 終了時刻 END) AS "11時台",
       MAX(CASE WHEN 開始時刻 = 12 THEN 終了時刻 END) AS "12時台",
       MAX(CASE WHEN 開始時刻 = 13 THEN 終了時刻 END) AS "13時台",
       MAX(CASE WHEN 開始時刻 = 14 THEN 終了時刻 END) AS "14時台",
       MAX(CASE WHEN 開始時刻 = 15 THEN 終了時刻 END) AS "15時台",
       MAX(CASE WHEN 開始時刻 = 16 THEN 終了時刻 END) AS "16時台",
       MAX(CASE WHEN 開始時刻 = 17 THEN 終了時刻 END) AS "17時台",
       MAX(CASE WHEN 開始時刻 = 18 THEN 終了時刻 END) AS "18時台"
  FROM
    (SELECT r.ReserveDate AS 予約日,
            s.StylistNo AS 担当スタイリスト番号,
            s.StylistName AS スタイリスト名,
            EXTRACT(hour from StartTime) AS 開始時刻,
            StartTime + CAST
              (ServiceTime || 'minutes' AS interval) AS 終了時刻
       FROM Reservation r
  RIGHT JOIN Stylist s ON r.StylistNo = s.StylistNo) t
    GROUP BY 予約日, 担当スタイリスト番号, スタイリスト名
    ORDER BY CASE WHEN 予約日 IS NULL THEN 1 ELSE 0 END,
            予約日, 担当スタイリスト番号
```

※ 設問 9･10 は、「10 時台」などの数字から始まる列タイトルを表記するために、ダブルクォーテーションを付けた形式で別名を指定しています。