基本問題の解答

題材A　銀行口座データベース

【01】 SELECT 口座番号 , 名義 , 種別 , 残高 , 更新日 FROM 口座
【02】 SELECT 口座番号 FROM 口座
【03】 SELECT 口座番号 , 残高 FROM 口座
【04】 SELECT * FROM 口座
【05】 UPDATE 口座 SET 名義 = ' ＸＸＸＸＸ '
【06】 UPDATE 口座 SET 残高 = 99999999, 更新日 = '2018-03-01'
【07】 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, '2018-03-13'); 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, '2017-11-30'); 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);
【08】 DELETE FROM 口座
【09】 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 < '2018-01-01'
【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 '2018-01-01' AND '2018-01-31'
【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 >= '2017-12-30' AND 更新日 < '2018-01-05'
【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
【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 FETCH FIRST 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 FETCH FIRST 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, '2018-04-01'); 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, '2018-04-02'); 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, '2018-04-03');

【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 < '2017-01-01'

【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 >= '2018-01-01'

【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 < '2018-01-01'

【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) - SUM(出金額), 0) FROM 取引 WHERE 口座番号 = '0351333' AND 日付 = '2018-01-11'), 更新日 = '2018-01-11' 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 = '2017-12-28') AS 入金額合計 , (SELECT SUM(出金額) FROM 取引 WHERE 口座番号 = '1115600' AND 日付 = '2017-12-28') 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 = '2016-03-01'

【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 = '2016-03-01'

【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 T. 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 >= '2018-01-01'

【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 >= '2018-01-01'

/* 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 >= '2018-01-01') 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

題材 B 商店データベース

【01】 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

【02】 SELECT 商品名 FROM 商品

【03】 SELECT * FROM 注文

【04】 SELECT 注文番号 , 注文枝番 , 商品コード FROM 注文

【05】 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');

【06】 SELECT * FROM 商品 WHERE 商品コード = 'W1252'

【07】 UPDATE 商品 SET 単価 = 500 WHERE 商品コード = 'S0023'

【08】 SELECT * FROM 商品 WHERE 単価 <= 1000

【09】 SELECT * FROM 商品 WHERE 単価 >= 50000

【10】 SELECT * FROM 注文 WHERE 注文日 >= '2018-01-01'

【11】 SELECT * FROM 注文 WHERE 注文日 < '2017-12-01'

【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 >= '2018-03-01' AND 注文日 < '2018-04-01' 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 >= '2018-03-01' 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 FETCH FIRST 10 ROWS ONLY

【29】 SELECT * FROM 商品 ORDER BY 単価 , 商品区分 , 商品コード OFFSET 5 FETCH FIRST 15 ROWS ONLY

【30】 SELECT * FROM 廃番商品 WHERE 廃番日 >= '2016-12-01' AND 廃番日 < '2017-01-01' 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 >= '2018-03-12' AND 注文日 < '2018-03-15' 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 = '201802250126' 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 >= '201710010001' AND 注文番号 <= '201710319999'

【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 FETCH FIRST 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)

【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 = '2018-03-15' AND 注文番号 = '201803150014' 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 = '2018-03-21' AND 注文番号 = '201803210080' 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 = '2018-03-22' AND 注文番号 = '201803220901' 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 = '201801130115' 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 > ='2016-08-01' AND T. 注文日 < '2016-09-01'

【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 > ='2016-08-01' AND T. 注文日 < '2016-09-01'

【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 , COALESCE(S. 商品コード || ':' || S. 商品名 , T. 商品コード || ':(廃番

済み)') AS 商品 , COALESCE(T. 数量 , 0) AS 数量 FROM 注文 AS T FULL 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'
/* FULL JOIN が使えない場合、以下で代替 */
SELECT 注文日 , 商品 , 数量 FROM (SELECT S. 商品区分 , T. 注文日 , COALESCE(S. 商品コード || ':' || S. 商品名 , T. 商品コード || ':(廃番済み) ') AS 商品 , COALESCE(T. 数量 , 0) AS 数量 FROM 注文 AS T LEFT JOIN 商品 AS S ON T. 商品コード = S. 商品コード UNION SELECT S. 商品区分 , T. 注文日 , COALESCE(S. 商品コード || ':' || S. 商品名 , T. 商品コード || ':(廃番済み)') 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 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 = '201704030010'

【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. 関連商品コード = S2. 商品コード

題材C RPG データベース

【01】 SELECT ID, 名称 , 職業コード , HP, MP, 状態コード FROM パーティー

【02】 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

【03】 SELECT * FROM イベント

【04】 SELECT イベント番号 AS 番号 , イベント名称 AS 場面 FROM イベント

【05】 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');

【06】 SELECT * FROM パーティー WHERE ID = 'C02'

【07】 UPDATE パーティー SET HP = 120 WHERE ID = 'A01'

【08】 SELECT ID, 名称 , HP FROM パーティー WHERE HP < 100

【09】 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 FETCH FIRST 3 ROWS ONLY

【29】 SELECT * FROM パーティー ORDER BY MP DESC OFFSET 2 FETCH FIRST 1 ROWS ONLY

【30】 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

【31】 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

【32】 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'

【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 <ALL (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' WHERE 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, (SELECT MAX(ルート番号) + 1 FROM 経験イベント));

【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

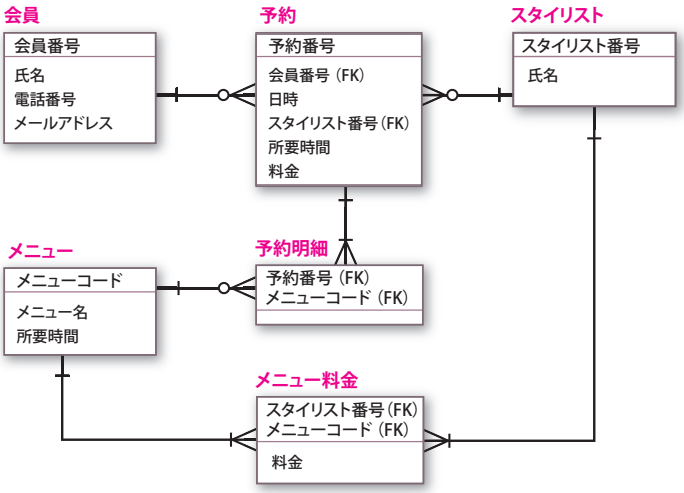
【69】 SELECT E1. イベント番号 , E1. イベント名称 , E1. 前提イベント番号 , E2. イベント名称 AS 前提イベント名称 , E1. 後続イベント番号 , E3. イベント名称 AS 後続イベント名称 FROM イベント E1 LEFT JOIN イベント E2 ON E1. 前提イベント番号 = E2. イベント番号 LEFT JOIN イベント E3 ON E1. 後続イベント番号 = E3. イベント番号 WHERE E1. 前提イベント番号 IS NOT NULL OR E1. 後続イベント番号 IS NOT NULL

【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

発展問題の解答

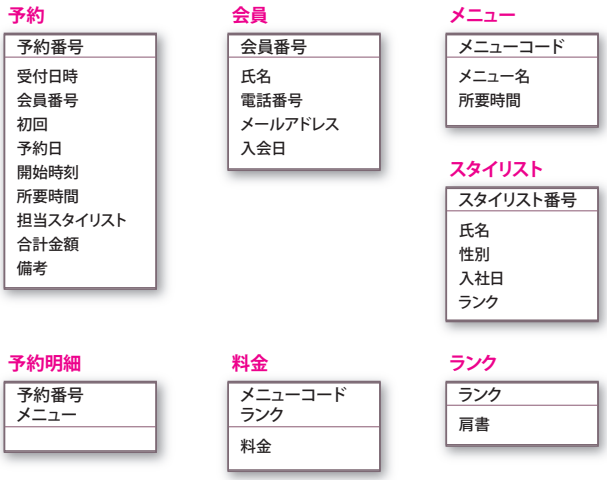
題材 D　ヘアサロンデータベース

問題 D-1-1 の解答



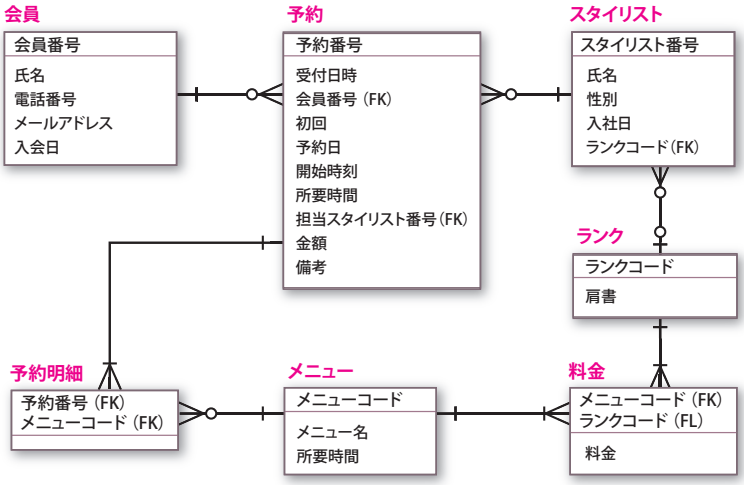
この問題のように、業務ルールからエンティティや属性、リレーションシップを導く方法を**トップダウン・アプローチ**といいます。このアプローチでは、業務ルールに登場するキーワードのみを対象に ER 図を作成します。このように取りまとめた図を**概念データモデル**と呼ぶこともあります。

問題 D-1-2 の解答



この問題のように、現場で業務に利用している各種帳票などを入手し、それを分析する手法を**ボトムアップ・アプローチ**といいます。このアプローチでは、業務ルールからだけでは拾いきれない細かな項目を整理することができます。

問題 D-1-3 の解答



業務ルールには現れなかった属性をシートから読み取り、各エンティティに追加します。また、メニューの料金はスタイリストに直接紐付くのではなく、スタイリストが属するランクによって決められていることを反映します。

このように取りまとめた図を**論理データモデル**と呼ぶこともあります。

問題 D-1-4 の解答

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	性別	Gender			○	CHAR	1		F：女性　M：男性
4	入社日	HireDate			○	DATE			
5	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、スタイリストの性別や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。

なお、問題文に掲載した物理設計図は、**物理データモデル**と呼ばれることもあります。

問題 D-1-5 の解答

```
-- 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,
  Gender      CHAR(1)     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。

問題 D-2-1 の解答

```
-- Member
INSERT INTO Member VALUES ('0001', '吉田康子', '0901112215',
'yoshida@a1.com', '2000-04-10');
INSERT INTO Member VALUES ('0002', '荒木和子', '0901112216',
'araki@a2.com', '2012-08-11');
INSERT INTO Member VALUES ('0003', '下田正一', '0901112217',
'shimoda@a3.com', '2013-04-12');
INSERT INTO Member VALUES ('0004', '風間由美子', '0901112218',
NULL, '2013-06-13');
INSERT INTO Member VALUES ('0005', '秋山美奈', '0901112219',
'akiyama@a5.com', '2015-01-14');
INSERT INTO Member VALUES ('0006', '木下博之', '0901112220',
'kinoshita@a6.com', '2015-04-15');
INSERT INTO Member VALUES ('0007', '広瀬正隆', NULL, NULL, '2016-
09-16');
INSERT INTO Member VALUES ('0008', '斉藤美紀', '0901112222',
'saitou@a8.co m', '2018-04-17');

-- 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', 'F', '1998-04-01',
'A');
INSERT INTO Stylist VALUES ('02', '佐藤茜', 'F', '2000-06-01',
'B');
INSERT INTO Stylist VALUES ('03', '井上博之', 'M', '2003-01-08',
```

```
'B');
INSERT INTO Stylist VALUES ('04', '小島正', 'M', '2010-05-02',
'C');
INSERT INTO Stylist VALUES ('05', '山田雄介', 'M', '2015-04-01',
'C');
INSERT INTO Stylist VALUES ('06', '市川紀子', 'F', '2018-06-10',
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, '2018-09-06 16:28', '0002',
'0', '2018-10-01', '17:00', 90, '01', 21600, NULL);
```

```
INSERT INTO Reservation VALUES (2, '2018-09-26 12:42', '0004',
'1', '2018-10-01', '10:00', 30, '03', 10000, NULL);
INSERT INTO Reservation VALUES (3, '2018-09-30 10:30', '0008',
'0', '2018-10-01', '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	2000-04-10
0002	荒木和子	0901112216	araki@a2.com	2012-08-11
0003	下田正一	0901112217	shimoda@a3.com	2013-04-12
0004	風間由美子	0901112218	(NULL)	2013-06-13
0005	秋山美奈	0901112219	akiyama@a5.com	2015-01-14
0006	木下博之	0901112220	kinoshita@a6.com	2015-04-15
0007	広瀬正隆	(NULL)	(NULL)	2016-09-16
0008	斉藤美紀	0901112222	saitou@a8.com	2018-04-17

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	Gender	HireDate	RankCD
01	秋葉ちか	F	1998-04-01	A
02	佐藤茜	F	2000-06-01	B
03	井上博之	M	2003-01-08	B
04	小島正	M	2010-05-02	C
05	山田雄介	M	2015-04-01	C
06	市川紀子	F	2018-06-10	(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	2018-09-06 16:28	0002	0	2018-10-01	17:00	90	01	21600	(NULL)
2	2018-09-26 12:42	0004	0	2018-10-01	12:00	30	03	10000	(NULL)
3	2018-09-30 10:30	0008	0	2018-10-01	15:00	150	05	26400	(NULL)

ReserveDetail

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

問題 D-2-2 の解答

```
1.   SELECT s.StylistName AS 名前,
            CASE s.Gender WHEN 'F' THEN '女性'
                          WHEN 'M' THEN '男性' END AS 性別,
            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, '2018-10-01 10:03',
   '0006', '0', '2018-10-01', '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, '2018-10-01 11:45',
    '0008', '0', '2018-10-02', '11:00', 90, '04', 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, startTime AS 開始時刻,
             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, startTime AS 開始時刻,
                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, startTime AS 開始時刻,
               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
```