問３　データベースの設計と運用（データベース）　(H24春-FE 午後問3)

【解答】
［設問１］　イ
［設問２］　エ
［設問３］　a－エ，b－ア
［設問４］　ウ

【解説】
データベースの設計とSQL文に関する問題である。設問１と設問３は，各要望に対応して表示する列と，各表に格納されている列の関係を把握すれば解答できる。表示する列と各表の列について，一つずつ確実に調べればよい。

設問２と設問４のSQL文は，出題頻度が高いグループ化のSQL文である。文法を理解して問題を解くことで，実力を向上させたい。また，解答群にある副問合せも出題頻度が高いので，グループ化と同様，きっちり理解する必要がある。

［設問１］
各要望に対応するため，最低限表示する列と，そのために必要な表の列の関係をまとめると次のようになる。

要望	表示する列	必要な表の列
要望１	社員番号，日付，料理名，皿数，単価	精算表：社員番号，日付， 精算コード（明細表との結合で使用） 明細表：皿数， 精算コード（精算表との結合で使用）， 料理コード（料理表との結合で使用） 料理表：料理名，単価， 料理コード（明細表との結合で使用）
要望２	日付，精算額合計	精算表：日付，精算額
要望３	料理名	料理表：料理名
要望４	日付，料理コード，皿数合計	精算表：日付， 精算コード（明細表との結合で使用） 明細表：料理コード，皿数， 精算コード（精算表との結合で使用）

したがって，要望２の（イ）が正解である。

［設問２］
社員ごとに“肉じゃが”の購入皿数を求めるグループ化のSQL文である。一人の社員が“肉じゃが”を複数回購入することがあるため，社員ごとに集計する。グループ化のSQL文の構文は，次のようになる。

SELECT句の指定した列名とは，GROUP BY句で指定した列名という意味である。言い方を変えるとグループごとの値（グループを代表する値）ということができる。もちろん，集合関数もグループごとの値である。

```
SELECT 指定した列名，集合関数
    FROM 表名
    WHERE 結合条件，抽出条件
    GROUP BY 指定した列名
```

したがって，（エ）が正解である。ここで，（エ）の記述４行の上２行は，精算表，明細表，料理表の結合条件を，３行目は，“肉じゃが”の抽出条件を示している。

ア：副問合せの結果，“肉じゃが”の料理コードが抽出される。そして，それと等しい明細表の行が抽出される。ここで，主問合せのFROM句から料理表を除き，GROUP BY句を追加すれば，正しいSQL文になる。FROM句に料理表を指定したにもかかわらず，料理表との結合条件がないと直積演算になる。

イ：HAVINGは，グループ化した後の抽出条件を指定する記述である。このSQL文では，グループ化した後の列名（SELECT句で指定した列名）に料理名がないためエラーになる。SELECT句とGROUP BY句に料理名を追加すれば正しい結果になる。

ウ：結合条件と抽出条件は正しいが，GROUP BY句が抜けている。この場合，SUM(明細表.皿数)によって，社員ごとではなく，全社員の皿数の総合計を求めることになる。また，全社員の皿数の総合計を集計するため，SELECT句には社員番号を記述できない。

＜副問合せについての補足＞
副問合せとは，WHERE句中に記述したSELECT文である。一方，元のSELECT文を主問合せという。このように，SELECT文が入れ子構造になっている。ここでは，副問合せのSELECT文を先に単独処理した後，副問合せで抽出された値を使って，主問合せのSELECT文を処理する。

＜直積演算についての補足＞
直積演算とは，二つの表のそれぞれの行同士を組み合わせて連結する演算である。二つの表の結合条件がないため，全ての行同士が連結される。例えば，３行の表と４行の表を直積演算した結果は，3×4＝12行の表になる。そのため，解答群（ア）の場合は，料理コードが異なる明細表の行と料理表の行も連結されてしまう。

［設問３］
設問１と同じく，最低限表示する列と，そのために必要な表の列を考える。
表示する列：社員番号，ある期間における１回の精算当たりの平均カロリー
必要な表の列：
精算表：社員番号，日付，精算コード（明細表との結合で使用）
明細表：皿数，精算コード（精算表との結合で使用），
　　料理コード（料理表との結合で使用）
料理表：カロリー，料理コード（明細表との結合で使用）
平均カロリーを求めるには，まず社員ごとに，“明細表の皿数×料理表のカロリー”を集計して，社員ごとの合計カロリーを求める。次に，それを社員ごとの精算回数で割ればよい。したがって，空欄ａは（エ）が正解である。

ここで，精算ごとの“明細表の皿数×料理表のカロリー”を集計した合計カロリーの列を精算表に追加すれば，精算表の社員番号，日付，合計カロリーだけで，求める列を全て表示できるため，明細表と料理表は不要になる。したがって，空欄ｂは（ア）が正解である。

［設問４］
料理ごとの販売皿数を求めるグループ化のSQL文である。また，料理表と明細表を結合することで，料理名とカロリーも表示している。更に，ORDER BY句で販売皿数の多い順に並び替えている。

解答群の各SQL文の違いは，次の２点である。１点目は，集合関数として（ア）と（イ）はCOUNTを，（ウ）と（エ）はSUMを記述している点である。２点目は，（ア）と（ウ）はWHERE句で料理表と明細表の結合条件を記述しているのに対し，（イ）と（エ）は結合条件を記述せず，副問合せを使用している点である。

SUM(明細表.皿数)は，皿数の値の合計を示す。一方，COUNT(明細表.皿数)は，行数の合計を示し，皿数の値は関係しないため，COUNT(*)としても同じ結果になる。なお，厳密にはCOUNT(明細表.皿数)とCOUNT(*)は，明細表.皿数にNULLがあるとき，COUNT(明細表.皿数)はNULLの行数をカウントしないので同じにはならない。この場合は，明細表.皿数にNULLはないのでCOUNT(明細表.皿数)とCOUNT(*)は同じ結果になる。通常，行数のカウントにCOUNT(*)を使用するのは，NULLの考慮をしなくてもよいためである。ここで，明細表の皿数が全て１であれば，SUM(明細表.皿数)とCOUNT(明細表.皿数)は同じ値になる。しかし，１回の精算に対し，同じ料理を２皿以上購入することもあるため，COUNTは不適切である。

料理表の行と明細表の行を料理コードで結合すると，同じ料理コードをもつ二つの行を合わせた行が生成される。それを料理ごとにグループ化して，皿数の合計を求めれば正しい結果になる。したがって，SUMと結合条件を記述した（ウ）が正解である。

（イ）と（エ）の副問合せは，明細表の皿数がNULLではない料理コードを求めている。ここで，明細表には販売した料理を記録するため，皿数がNULLになることはなく，販売した料理の料理コードが抽出される。これは，皿数の合計を求めるために必要ない処理である。また，主問合せのFROM句に料理表を指定したにもかかわらず，料理表と明細表との結合条件がないため，直積演算になってしまう。

＜集合関数についての補足＞
集合関数には，SUMとCOUNTのほかに，次の三つがある。
AVG(列名)：指定した列の値の平均
MAX(列名)：指定した列の値の最大
MIN(列名)：指定した列の値の最小

＜ORDER BY句についての補足＞
DESCは，降順に並べる指定である。ASCを指定，又は指定を省略した場合は，昇順に並び替える。ORDER BY句で指定できる列名は，SELECT句で指定した列名である。