Exercice 1

Soit la base de données BANQUE contenant les tables suivantes :

* AGENCE (\*Num\_Agence, Nom, Ville, Actif)
* CLIENT (\*Num\_Client, Nom, Prenom, Ville)
* COMPTE (\*Num\_Compte, Num\_Agence#, Num\_Client#, Solde)
* EMPRUNT (\*Num\_Emprunt, Num\_Agence#, Num\_Client#, Montant)
1. Que représentent les symboles étoile (\*) et astérisque (#) ?
2. Sans utiliser DISTINCT, donnez une requête équivalente en SQL :

SELECT DISTINCT Num\_Client

FROM COMPTE

WHERE solde < 1000

OR solde > 100000 ;

Exercice 2 :

Dans la situation de l’exercice précédent, écrivez les requêtes SQL correspondant aux questions suivantes : :
(a) Les clients n’ayant pas de compte dans la même agence que Liliane Bettencourt.
(Tableau résultat : Num\_Client).

(b) Les agences ayant un actif plus élevé que toutes les agences de Saint-Ouen. (Tableau
résultat : Num\_Agence)

(c) Le solde moyen des comptes clients, pour chaque agence dont le solde moyen est
supérieur à 10000. (Tableau résultat : Num\_Agence, Solde\_Moyen)

(d) Le nombre de clients de l’agence de nom “Paris-BNF” dont la ville n’est pas renseignée dans la relation CLIENT. (Tableau résultat : Nombre).

(e) Les clients ayant un compte dont le solde est supérieur à la somme totale de tous
les actifs des agences de Saint-Ouen. (Tableau résultat : Num\_Client).

(f) Les clients dont la somme du solde de tous les comptes est inférieure à l’actif de
chaque agence. (Tableau résultat : Num\_Client)

(g) Les clients ayant un compte dans toutes les agences de Saint-Ouen. (Tableau résultat : Num\_Client).

**Réponses aux exercices**

Exercice 1 :

1)Les clefs primaires sont précédées d’une étoile (\*) et les clefs étrangères sont suivies d’un astérisque (#).

2)Une première solution exploite le fait que l’opérateur d’union est un opérateur ensembliste et élimine donc les doublons :

(SELECT Num\_Client

FROM COMPTE

WHERE solde < 1000)

UNION
(SELECT Num\_Client

FROM COMPTE

WHERE solde > 100000) ;

Une seconde solution utilise le GROUP BY (utilisé d’ordinaire dans le cadre des requêtes d’agrégation) :

SELECT Num\_Client

FROM COMPTE

WHERE solde < 1000

OR solde > 100000

GROUP BY Num\_Client ;

 Une troisième solution exploite le fait que Num\_Client est clef primaire de client :

SELECT Num\_Client

FROM CLIENT

WHERE Num\_Client IN

(SELECT Num\_Client

FROM COMPTE

WHERE solde < 1000

OR solde > 100000) ;

Exercice 2 :

1. Attention, la question est plus difficile qu’il n’y parait. Malgré les apparences la
requête suivante par exemple ne convient pas :

SELECT Num\_Client

FROM COMPTE

WHERE Num\_Agence NOT IN

(SELECT Num\_Agence

FROM COMPTE NATURAL JOIN CLIENT

WHERE Client.Nom=‘Bettencourt’

AND Client.Prenom=‘Liliane’) ;

Cette requête retourne les clients qui ont un compte dans une agence dans laquelle
Liliane Bettencourt n’a pas de compte. En fait, il fallait écrire quelque chose de plus compliquée, par exemple :

SELECT Num\_Client

FROM Client

EXCEPT
(Select Num\_Client

FROM Compte

WHERE Num\_Agence IN

(SELECT Num\_Agence

FROM COMPTE NATURAL JOIN CLIENT

WHERE Client.Nom=‘ Bettencourt’

AND Client.Prenom= ‘Liliane’)) ;

ou bien

SELECT Num\_Client

FROM Client

WHERE Num\_Client NOT IN

(Select Num\_Client

FROM Compte

WHERE Num\_Agence IN

(SELECT Num\_Agence

FROM COMPTE NATURAL JOIN CLIENT

WHERE Client.Nom=‘ Bettencourt’

AND Client.Prenom= ‘Liliane’)) ;

1. SELECT Num\_Agence

FROM Agence

WHERE Actif > ALL

(SELECT Actif

FROM Agence

WHERE Ville=‘Saint Ouen’ ) ;

Attention, utiliser > ANY au lieu de > ALL sélectionne les agences ayant un actif plus
élevé qu’au moins une agence de Saint Ouen.

Une solution équivalente à celle utilisant > ALL :

SELECT Num\_Agence

FROM Agence

WHERE Actif >

(SELECT Max(Actif)

FROM Agence

WHERE Ville=‘Saint Ouen’ ) ;

Attention, la syntaxe .... > MAX (SELECT Actif FROM...) est incorrecte.

1. SELECT AVG(Solde) as Solde\_Moyen

FROM Compte

GROUP BY Num\_Agence

HAVING AVG(Solde) > 10000 ;

1. SELECT COUNT(DISTINCT num\_client) as Nombre

FROM Client, Compte, Agence

WHERE Client.Num\_client=Compte.Num\_client

AND Agence.Num-Agence=Compte.Num-Agence

AND Agence.Nom=‘Paris-BNF’

AND Client.Ville IS NULL ;

Attention à la jointure naturelle sur Client et Agence. La jointure sera entre autres
faite sur les deux attributs ville, ce qui forcera une contrainte supplémentaire sur
les données (si un client possède un compte dans une agence domiciliée dans une
ville différente de celle où il habite, alors ce compte n’apparaitra pas). Attention
également à ne pas oublier le mot clef Distinct : nous ne voulons compter chaque
client ayant un compte dans l’agence “Paris-BNF” qu’une seule fois, même s’il y
possède plusieurs comptes. Une solution alternative sans Distinct :

SELECT SUM(num\_client) as Nombre

FROM Client

WHERE Ville IS NULL

AND Num\_Client IN

(SELECT Num\_Client

FROM Compte NATURAL JOIN Agence

WHERE Agence.Nom=‘ Paris-BNF’) ;

1. SELECT Num\_Client

FROM Compte

WHERE Solde >

(SELECT SUM(Actif)

FROM Agence

WHERE Ville=‘Saint-Ouen’) ;

Si l’on veut éviter les doublons liés au fait qu’un même client pourrait avoir plusieurs
comptes satisfaisant cette propriété, on peut utiliser le mot clef DISTINCT :

SELECT DISTINCT Num\_Client

FROM Compte

WHERE Solde >

(SELECT SUM(Actif)

FROM Agence

WHERE Ville=‘Saint-Ouen’) ;

1. SELECT Num\_Client

FROM Compte

GROUP BY Num\_Client

HAVING SUM(Solde) <

(SELECT MIN(Actif)

FROM Agence) ;

1. Une première solution sans utiliser l’agrégation, mais avec double imbrication et
négations :

SELECT Num\_Client

FROM Client

WHERE NOT EXISTS

(SELECT \* FROM Agence

WHERE Ville=‘Saint-Ouen’

AND NOT EXISTS

(SELECT \* FROM Compte

WHERE Client.Num\_Client=Compte.Num\_Client

AND Compte.Num\_Agence=Agence.Num\_Agence));

Une autre solution sans négation, mais avec de l’agrégation :

SELECT Num\_Client

FROM Compte, Agence

WHERE Compte.Num\_Agence=Agence.Num\_Agence

AND Ville=‘Saint-Ouen’

GROUP BY Num\_Client

HAVING COUNT(DISTINCT Num\_Agence)=

(SELECT COUNT(DISTICT Num\_Agence)

FROM Agence

WHERE Ville=‘Saint-Ouen’);

Cette dernière solution est basée sur le fait que si un client a un compte dans
chacune des agences de Saint-Ouen et qu’il y a exactement *n* agences à Saint-Ouen,
alors ce client a un compte dans exactement *n* agences à Saint-Ouen.