Exo 01

malade(code_mal, nom, prenom, adresse, tel)
chirurg(code_ch, nom, specialite)
occupe(code_mal, date_deb, date_fin, num_lit)
opere(code_ch, code_mal, date_op, nature, heure)

1- Donner le code des chirurgiens qui sont travaillé le 09-09-01
2- Donner le nom et le prénom des personnes qui sont rentrée à l'hôpital le 05-03-01
3- Donner le nombre d'intervention de chaque chirurgien entre le 09-05-01 et 09-09-01
4- Le nom des chirurgiens ayant travaillé au moins une fois à la même date que le chirurgien 'M^{ed}'

Exo 02

Soit la base de données relationnelle suivante :
client(N°cl, nom_cl, adresse_cl)
article(N°art, nom_art, prix_achat_art, prix_vente_art, poids_art, coul_art)
commande(N°cde, N°cl, date_cde, N°four)
ligne_cde(N°cde, N°ligne, N°art, qte_cde)
fournisseur(N°four, nom_four, adresse_four)
Exprimer les requêtes suivantes en SQL :

1- Donner tous les articles pour lesquels le prix de vente est supérieur ou égal au double de prix d'achat
2- Donner tous les articles rouges de poids >100g
3- Pour obtenir les articles qui ne sont pas rouges et dont le poids est inférieur ou égal à 500g
4- Pour rechercher les fournisseurs qui pendant la période du 09-05-01 au 09-09-01 ont réalisé plus de 10 vente
5- Pour calculer le prix de vente moyen des articles de chaque couleur en excluant les articles pour lesquels le prix d'achat est inférieur à 500DA
6- Pour rechercher tous les articles dont le poids est inférieur au poids de l'article numéro'A02'
7- Pour rechercher dans la table articles, les articles de même couleur que l'article 'A10' et dont le poids est supérieur ou égal au poids moyen de tous les articles
8- On veut rechercher la liste des articles dont le prix de vente est supérieur au prix de vente de l'article de couleur blanche de moins chère

Exo 03

Soit la base de données suivantes :
boisson(N°bois, origine, date_fab, concentration)
buveur(N°buv, nom, adresse)
bus(N°buv, N°bois, quantité)
Exprimer en SQL :

1. Donner les origines et les numéros du boisson fabriqué en 09-09-2001, et concentration supérieur à 10 triés par ordre alphabétique croissant sur origine et décroissant sur numéro de boisson
2. Donner les noms des buveurs ayant bus un boisson d'Abricot
3. Donner les numéros de boisson bus par plus de 100 buveurs
4. Insérer le tuple (200, orange, 09-09-2001, 14) dans la relation boisson
5. Supprimer tous les boissons bus par 'Bendjima'
6. Mettre à jour les quantités bus à 0 pour tous les buveurs d'adresse Béchar

Exo 04

Soit la base de données relationnel suivantes :

transport(client, gare_origine, gare_destination, N°wagon, type_marchandise, poids_marchandise, date_chargement)
réseau(gare_origine, gare_destination, gare_suivante, N°ligne)
ligne(N°ligne, rang, gare)
trafic(N°train, N°ligne, jour)
train(N°train, N°wagon)
wagon(N°wagon, type_wagon, poids_vide, capacite, etat, gare)

Exprimer les requêtes suivantes en SQL :

1. Donner la liste des numéros de wagon de type 'frigo' disponible à la gare de tours
2. Donner la liste des types des wagons du train 4002
3. Construire une relation qui donne pour le train 4002 et pour chaque wagon, son numéro, le poids des marchandises et le poids à vide
4. Donner les numéros de lignes tels qu'il existe un train tous les jours
5. Donner les numéros de lignes tels qu'il existe un train lundi et jeudi
6. Construire une relation qui donne pour chaque numéro de train le nombre de wagons
7. donner les numéros de train ayant plus de 100 wagons
8. Quelle est la gare d'arrivé de la ligne 35

Exo 05

Soit la base de données de l'université :

etudiant(nom_etud, age_etud, ville)
enseignant(nom_ens, age_ens, nom_fac, grade, salaire)
faculte(nom_fac, localite, recteur)
affectation(nom_et, groupe)
emploi_temps(nom_ens, groupe, matiere, salle, jour, heure, activite_pedag)

1- déterminer les clés primaires et étrangères de ces relations
2- créer en SQL les relations : emploi_temps, faculte
3- exprimer en SQL les requêtes suivantes :
   Q1 :trouver le couple des noms différents des étudiants habitant la même ville.
   Q2 :trouver toutes les informations concernant les étudiants n'habitant pas la ville Béchar
   Q3 :trouver le nom et l'âge des étudiants affectés au groupe 3 et dont le nom se termine par la lettre 'B'
   Q4 :trouver le nom et le salaire des enseignants dont le salaire dépasse moyenne des salaires et dont l'âge est compris entre 35 et 50 ans
   Q5 :trouver les noms des enseignants de moins de 35 ans, de grade 'chargé de cours' qui enseignent la matière 'BDD' à la fac d'Oran dont le recteur est M^{ed} .

.
.

Exo 06

client(id_cl, nom, adresse, code_post, ville, tel)
article(id_art, designation, prix_unit, qte_stock)
commande(N°cde, id_cl, id_vend, date_cde)
ligne_cde(N°cde, N°ligne, id_art, qte_cde)
vendeur(id_vend, nom_vend, quantite, salaire)

1- Pour visualiser tout les données concernant les clients
2- On désire connaître seulement le nom et le numéro de téléphone des clients
3- Pour visualiser uniquement les clients qui habitent à Béchar
4- Lister tous les articles dont le prix unitaire est > à 150DA et dont quantité en stock est ≤ à 100 et toutes les commandes enregistrés après la date 09-09-99
5- Lister tous les articles dont le prix est compris entre 150 et 200 ainsi que toutes les commandes non enregistrer entre la date 09-09-99 et 30-01-00
6- Lister tous les clients des villes 'Béchar' 'Oran' 'Alger' ainsi que tous les articles dont le prix unitaire est 150, 200 ou 300

Exo 07

hôtels (nom_hot, ville_hot, lits_dip, date)
vols (N°vol, compagnie, ville_dep, ville_arr, date_dep, date_arr, places_disp)
voyages (N°voyage, N°vol_all, N°vol_ret, nom_hot)
Exprimer en SQL les requêtes suivantes :

1. Les numéros de voyage et le et le nombre de places disponibles dans les vols aller et retour pour chaque voyage.
2. Le nombre d'hôtels à Béchar.
3. le nombre de lits disponibles à Béchar le 09-09-2002
4. Le nombre de lits disponible pour le voyage 113 (les lits sont limitées par le nombre de lits disponibles dans l'hôtel pendant la période du voyage).
5. Toutes les villes dans la table hôtels et pour chaque ville le nombre d'hôtels.

Exo 08

Soit les trois relation suivantes :

```
CREATE TABLE fournisseur
(N°four  char(5) not null,
nom_four char(20) , status integer,
ville char(10) ,
PRIMARY KEY(N°four)) ;

CREATE TABLE produit
(N°prod  char(6) not null,
nom_prod char(20) ,  couleur char(6),
poids integer,
PRIMARY KEY(N°prod)) ;
```

```
CREATE TABLE client
(N°cl  char(6) not null,
nom_cl char(20) ,
ville char(10) ,
PRIMARY KEY(N°cl)) ;

CREATE TABLE commande
(N°four  char(5) not null,
N°prod  char(6) not null,
N°cl  char(6) not null,
Qte integer,
PRIMARY KEY(N°four , N°prod , N°cl)) ;
```

Exprimer les requêtes suivantes en SQL:

1. Toutes les informations sur les clients.
2. Toutes les informations sur les clients à 'Béchar'.
3. La liste triée des numéros des fournisseurs du client avec le numéro C1.
4. Les commandes avec une quantité entre 300 et 750.
5. Les commandes avec une quantité différente de NULL.
6. Les numéros des clients qui sont situés dans une ville qui commence par "B".
7. Les numéros des fournisseurs et des clients qui sont situés dans la même ville.
8. Les numéros des fournisseurs et des clients qui ne sont pas situés dans la même ville.
9. Les numéros des produits fournis par des fournisseurs à Béchar.
10. Les numéros des produits fournis par des fournisseurs 'Béchar' à des clients 'Oran'.
11. Les couples de villes (v_i , v_j) tel qu'il existe au moins un fournisseur dans la ville v_i d'un client dans la ville v_j .
12. Les numéros des produits fournis à des clients situés dans la même ville que leurs fournisseurs.

Exo 09

Soit le modèle relationnel suivant relatif à la gestion des notes annuelles d'une promotion d'étudiants :

etudiant (N°etud, nom_etud, prénom_etud)
matière (code_mat, libellé_mat, coeff_mat)
evaluer (N°etud, code_mat, date, note)

Exprimer les requêtes suivantes en SQL :

1. Quel est le nombre total d'étudiants
2. Quelles sont, parmi l'ensemble des notes, la note la plus haute et la note la plus basse
3. Quelles sont les moyennes de chaque étudiant dans chacune des matières
4. Quelles sont les moyennes par matière
5. Quelle est la moyenne générale de chaque étudiant
6. Quelle est la moyenne générale de la promotion
7. Quels sont les étudiants qui ont une moyenne générale supérieure ou égale à la moyenne générale de la promotion

Exo 10
Soient la bases de données suivante :
Avion (N°avion , marque , nbr_places)
Pilote (N°pilote , nom_pil , adresse_pil)
Vol (N°vol , date_vol , N°avion , N°pilote , ville_d , ville_a ,heure_d , heure_a)

Exprimer en **SQL** les requêtes suivantes :

1. Donnez le nombre de pilotes ‘Bécharien’
2. Donnez le nombre de places pour tous les avions
3. Donnez tous les vols effectués par des pilotes ‘Bécharien’
4. Donnez les noms des pilotes qui ont pilotés tous les avions.

Exercice (supplémentaire)
Soit la relation suivante :
transport(N°train, N°wagon, type_wagon, poids_vide, capacité, état, gare, gare_destination, N°ligne, rang_ligne, date)
Soit les dépendances fonctionnelles les suivantes :
N°wagon → capacité ; N°wagon → type_wagon ; N°wagon → poids_vide ;
N°wagon → etat ; N°wagon → gare_destination
N°ligne → gare ; N°ligne → rang_ligne ;
N°train → N°ligne ; N°train → date

- La relation transport est elle en 3FN ? sinon décomposée la ?
- Exprimer en SQL :
  1. Donner la liste des lignes qui partent de la gare d'Alger rangé par ordre croissant
  2. Donner pour chaque wagon sa capacité, et son poids vide
  3. Donner pour chaque wagon du train 2300, son numéro, sa capacité, et son poids vide
  4. Donner la liste des numéros de wagon qui partent d'Oran le 09-09-01 et leurs capacité
  5. Donner les numéros de wagons chargés du train 4002 ainsi que leur type

Exercice (supplémentaire)
Soit la base de données suivantes
buveur(code_buv, nom_buv, ville)
boisson(code_boiss, nom_boiss, couleur, degré, année)
bus(code_buv, code_boiss, quantité)
producteur(code_prod, nom_prod, région)
produire(code_prod, code_boiss, quantite)

Créer en SQL les relations :boisson, bus
Exprimer en SQL les requêtes suivantes :
1. Donner les noms de boisson sans double
2. Donner les buveurs ayant bu de boisson dont le nom commence par 'B', degré inconnu
3. Donner les noms des boissons bus par au moins un buveur
4. Calculer la moyenne des degrés pour chaque boisson
5. Insérer les producteurs de la région de Béchar ayant produit des boissons de 2001 dans la relation buveur

Exercice (supplémentaire)
enseignant(nom_ens, age, grade, sal, indice, nom_fac, chef)
etudiant(nom_etud, age, ville)
fac(nom_fac, localité, moyen)
affectation(nom_etud, groupe)
appart_fac(groupe, nom_fac)

Exprimer les requêtes suivantes en SQL :
1. Trouver le nom de la fac a qui appartient l'étudiant 'Omar'
2. Trouver les salaires de tous les enseignants
3. Trouver le salaire et l'âge pour l'enseignant M^{ed} âgé de 40 ans

Exercice (supplémentaire)
livre (code_livre, titre_livre, editeur, année_edition, prix_achat, date_achat)
auteur (N°auteur, nom_auteur, prénom_auteur)
cassette (N°cassette, état, durée_enreg, durée_disp, prix_achat, date_achat)
film (N°film, titre_film, categorie, N°cassette)
emprunteur (N°emp, nom_emp, prénom_emp, adresse_emp)
ecrire (N°auteur, code_livre)
emprunter_livre (N°emp, code_livre, date_emp, date_retour)
emprunter_film (N°emp, N°film, date_emp, date_retour)

Ecrire en SQL, les requêtes permettant d'obtenir les réponses aux questions suivantes:
1. quelle sont les titres et catégories des films enregistrés sur la cassette numéro 3 ? les titres seront rangés par ordre alphabétique
2. quels sont les livres qui ont été empruntés et combien de fois l'on-ils été ? en résultat :les titres et le nombre d'emprunts rangés par ordre décroissant du nombre d'emprunts et pour un même nombre, par ordre alphabétique des titres
3. quels films ont été empruntés plusieurs fois par une même personne ? le résultat fournira le numéro d'emprunteur, son nom, son prénom et le titre du film
4. quel est le prix moyen des cassettes préenregistrées ?
5. quelles sont les cassettes disponibles ? en résultat on donnera les numéros des cassettes rangés dans l'ordre croissant et pour chacun la liste des titres des films qu'elle contient (rangés par ordre alphabétique). Le numéro des cassettes ne sera pas répété.

Exercice (supplémentaire)
Soit la base de données relationnelle suivantes :
Département (N°dep, nom_dep, budget_dep, directeur)
Employé (N°emp, nom_emp, N°dep, N°tel, ville)
Projet (N°proj, titre, budget_proj, N°dep)
Historique (N°proj, N°emp, date, salaire)
1. Déterminer les clés primaires et les clés étrangères de ces relations.
2. crées en **SQL** les relations PROJET et HISTORIQUE.
3. Exprimer en **SQL** les requêtes suivantes :
   - donner les numéros des employés habitant dans la même ville que l'employé N°113
   - donner les villes des employés ayant participer au projet 'Bases de données'
   - donner le budget le plus élevé des projets du département N°09
   - supprimer le projet 'intelligence artificielle'.

Exercice (supplémentaire)

Etudiant

nom	ville	age
Reda	Alger	18
Omar	Alger	19
Méliani	Béchar	20

Stagiaire

nom	ville	age
Dalila	Alger	21
Fatiha	Alger	17
Omar	Béchar	20

Donner les résultats des requêtes **SQL** suivantes :

- SELECT nom FROM etudiant
- SELECT age FROM etudiant WHERE ville = ‘Béchar’
- SELECT nom , ville FROM stagiaire WHERE age>21
- SELECT nom FROM stagiaire
  INTERSECT
  SELECT nom FROM etudiant
- SELECT count(*) FROM etudiant

Sol. exo 01

1- SELECT code_ch
FROM opere
WHERE date_op='09-09-01'

2- SELECT nom, prenom
FROM malade
WHERE code_mal IN (SELECT code_mal
FROM occupe
WHERE date_deb='05-03-01')

3- SELECT code_ch, COUNT(*)
FROM chirurg, opere
WHERE chirurg.code_ch=opere.code_ch
AND date_op BETWEEN 09-05-01 AND 09-09-01
GROUP BY code_ch

4- SELECT nom
FROM opere, chirurg
WHERE chirurg.code_ch=opere.code_ch
AND date_op IN (SELECT date_op
FROM chirurg, opere
WHERE chirurg.nom='M^{ed},
AND chirurg.code_ch=opere.code_ch)

Sol. exo 02

1- SELECT *
FROM article
WHERE prix_vente_art >= 2*prix_achat_art

2- SELECT *
FROM article
WHERE coul_art = 'rouge'
AND poids_art > 100

3- SELECT *
FROM article
WHERE coul_art ≠ rouge AND pods_art ≤ 500

4- SELECT N°four
FROM commande
WHERE 09-05-01 ≤ date_cde ≤ 09-09-01
GROUP BY N°four
HAVING COUNT(*) > 10

5- SELECT coul_art, AVG(prix_vente_art)
FROM article
WHERE prix_achat_art ≥ 500
GROUP BY coul_art

6-
```
SELECT *
FROM article
WHERE poids_art < (SELECT poids_art
                   FROM article
                   WHERE N°art='A02')
```

7-
```
SELECT N°art, nom_art
FROM article
WHERE coul_art = (SELECT coul_art
                  FROM article
                  WHERE N°art='A10')
AND poids_art ≥ (SELECT AVG(poids_art)
                 FROM article)
```

8-
```
SELECT nom_art
FROM article
WHERE prix_vente_art>(SELECT MIN(prix_vente_art)
                      FROM article
                      WHERE coul_art='blanche'
```

Sol. exo 03

1.
```
SELECT origine, N°bois
FROM boisson
WHERE date_fab='09-09-2001'
AND concentration>10
ORDER BY origine ASC, N°bois DESC
```

2.
```
SELECT nom
FROM buveur, bus
WHERE buveur.N°buv=bus.N°buv
AND bus.N°bois=boisson.N°bois
AND boisson.origine='Abricot'
```

3.
```
SELECT N°bois
FROM bus
GROUP BY N°bois HAVING COUNT(*)>100
```

4.
```
INSERT INTO boisson<200, orange, 09-09-2001, 14>
```

5.
```
DELETE boisson
WHERE 'Bendjima' IN (SELECT nom
                     FROM buveur, bus
                     WHERE boisson.N°bois=bus.N°bois
                     AND bus.N°buv=buveur.N°buv)
```

6. UPDATE bus
SET quantité=0
WHERE 'Béchar' IN (SELECT adresse
FROM buveur
WHERE bus.N°buv=buveur.N°buv)

Sol. exo 04

1- SELECT N°wagon
FROM wagon
WHERE type_wagon='frigo'
AND etat='libre'
AND gare='tours'

2- SELECT type_wagon
FROM wagon
WHERE N°wagon IN (SELECT N°wagon
FROM train
WHERE N°train='4002'

3- SELECT N°wagon, poids_vide, poids_marchandise
FROM train, wagon, transport
WHERE train.N°wagon = wagon.N°wagon
AND train.N°wagon = transport.N°wagon
AND train.N°train = '4002'

4- SELECT N°ligne
FROM trafic
GROUP BY N°ligne HAVING SET(jour) CONTAIN (SELECT jour
FROM trafic)

5- SELECT N°ligne
FROM trafic
GROUP BY N°ligne
HAVING SET(jour) CONTAIN ('lundi', 'jeudi')

Autre méthode :

SELECT N°ligne
FROM trafic
WHERE jour='lundi'
INTERSECT
SELECT N°ligne
FROM trafic
WHERE jour='jeudi'

6- SELECT N°train, COUNT(N°wagon)
FROM train
GROUP BY N°train

7- SELECT N°train
FROM train
GROUP BY N°train
HAVING COUNT(N°wagon) ≥ 100

8- SELECT gare
FROM ligne
WHERE rang = (SELECT MAX(rang)
FROM ligne
WHERE N°ligne='35')
AND N°ligne = '35'

Sol. exo 05

1- clés primaires :
etudiant ⇒ nom_etud
enseignant ⇒ nom_ens
faculte ⇒ nom_fac
affectation ⇒ nom_etud, groupe
emploi_temps ⇒ nom_ens, jour, heure
clés étrangères :
enseignant ⇒ nom_fac % à faculte
affectation ⇒ nom_etud % à etudiant
emploi_temps ⇒ nom_ens % à enseignant

2- création des tables en SQL :
CREATE TABLE faculte
(nom_fac char(20) not null,
localite char(20), recteur char(15))
PRIMARY KEY nom_fac ;

CREATE TABLE emploi_temps
(nom_ens char(15) not null,
groupe integer, matière char(10),
jour char(8) not null, heure char(15) not null,
activite_pedag char(15),
PRIMARY KEY(nom_ens, jour, heure),
FORCEGN KEY nom_ens, IDENTIFIES enseignant ;

3- Exprimer en SQL :
R1 :SELECT A.nom_etud, B.nom_etud
FROM etudiant A, etudiant B
WHERE A.ville = B.ville
AND A.nom_etud <> B.nom_etud

R2 :SELECT *
FROM etudiant
WHERE ville <> 'Béchar'

Autre méthode :

```
SELECT *
FROM etudiant
WHERE ville not IN (SELECT ville
                    FROM etudiant
                    WHERE ville='Béchar'
```

R3 :

```
SELECT nom_etud, age_etud
FROM etudiant
WHERE nom_etud LIKE '%B'
AND nom_etud IN (SELECT nom_etud
                 FROM affectation
                 WHERE groupe=4)
```

Autre méthode :

```
SELECT nom_etud, age_etud
FROM etudiant, affectation
WHERE nom_etud LIKE '%B'
AND groupe=4
AND etudiant.nom_etud=affectation.nom_etud
```

R4 :

```
SELECT nom_ens, salaire
FROM enseignant
WHERE age_ens BETWEEN 35 AND 50
AND salaire>(SELECT AVG(salaire)
             FROM enseignant)
```

R5 :

```
SELECT nom_ens
FROM emploi_temps
WHERE matiere='BDD'
AND nom_ens IN (SELECT nom_ens
                FROM enseignant
                WHERE grade = 'charge de cours'
                AND age_ens<35
                AND nom_fac IN (SELECT nom_fac
                                FROM faculte
                                WHERE  localite='ORAN'
                                AND recteur='M^ed,
```

Sol. exo 06

1.

```
SELECT *
FROM client
```

2.

```
SELECT nom, tel
FROM client
```

3.

```
SELECT *
FROM client
WHERE ville=’Béchar’
```

4.

```
SELECT *
FROM article
WHERE prix_unit>150DA
AND qte_stock ≤ 100
UNION
SELECT *
FROM commande
WHERE date_cde > 09-09-99
```

5.

```
SELECT *
FROM article
WHERE prix_unit ≥ 150
AND prix_unit ≤ 200
UNION
SELECT *
FROM commande
WHERE date_cde BETWEEN 09-09-99 AND 30-01-00
```

6.

```
SELECT *
FROM client
WHERE ville=’Béchar’ OR ville=’Oran’ OR ville=’Alger’
UNION
SELECT *
FROM article
WHERE prix_unit=150
OR prix_unit=200
OR prix_unit=300
```

Sol. exo 07

1. SELECT N°voyage , A.places_disp , B.places_disp
   FROM vol A , vol B , voyages
   WHERE A.N°vol = N°vol_all
   AND B.N°vol = N°vol_ret
2. SELECT COUNT(nom_hot)
   FROM hotels
   WHERE ville_hot = ‘Béchar’
3. SELECT SUM(lits_disp)
   FROM hotels
   WHERE ville_hot = ‘Béchar’
   AND date = ’09-09-2002’
4. SELECT MIN(lits_disp)
   FROM hotels , voyages
   WHERE hotels.nom_hot = voyages.nom_hot
   AND voyages.N°voyage = ‘113’
   AND hotels.date >= voyages.date_dep
   AND hotels.date <= voyages.date
5. SELECT ville_hot , COUNT(nom_hot)
   FROM hotels
   GROUP BY ville_hot

Sol. exo 08

1.
SELECT N°cl, nom_cl, ville
FROM client
ou
SELECT *
FROM client
2.
SELECT *
FROM client
WHERE ville = ’Béchar’
3.
SELECT DISTINCT N°four
FROM command
WHERE N°cl = ’C1’
ORDER BY N°four
4.
SELECT *
FROM commande
WHERE Qte >= 300 AND Qte <= 750
ou
SELECT *
FROM commande
WHERE Qte BETWEEN 300 AND 750
5.
SELECT *
FROM commande
WHERE Qte IS NOT NULL

ou

```
SELECT *
FROM commande
WHERE Qte = Qte
```

6.

```
SELECT N°cl
FROM client
WHERE ville LIKE 'B%'
```

ou

```
SELECT N°cl
FROM client
WHERE SUBSTR (ville, 1, 1) = 'B'
```

7.

```
SELECT N°four, N°cl
FROM fournisseur, client
WHERE fournisseur.ville = client.ville
```

8.

```
SELECT N°four, N°cl
FROM fournisseur, client
WHERE NOT fournisseur.ville = client.ville
```

ou

```
SELECT N°four, N°cl
FROM fournisseur, client
WHERE fournisseur.ville <> client.ville
```

9.

```
SELECT DISTINCT N°prod
FROM commande, fournisseur
WHERE commande.N°four = fournisseur.N°four
AND ville = 'Béchar'
```

10.

```
SELECT DISTINCT N°prod
FROM commande, fournisseur, client
WHERE commande.N°four = fournisseur.N°four
AND commande.N°cl = client.N°cl
AND fournisseur.ville = 'Béchar'
AND client.ville = 'Oran'
```

11.

```
SELECT DISTINCT fournisseur.ville, client.ville
FROM commande, fournisseur, client
WHERE commande.N°four = fournisseur.N°four
AND commande.N°cl = client.N°cl
```

12.

```
SELECT DISTINCT N°prod
FROM commande, fournisseur, client
WHERE commande.N°four = fournisseur.N°four
AND commande.N°cl = client.N°cl
AND client.ville = fournisseur.ville
```

Sol. exo 09

1. SELECT count(N°etud)
   FROM etudiant

2. SELECT max(note),min(note)
   FROM evaluer

3. SELECT libelle_mat, nom_etud, avg(coeff_mat*note)
   FROM etudiant, matiere, evaluer
   WHERE etudiant.N°etud=evaluer.N°etud
   AND matiere.code_mat=evaluer.code_mat
   GROUP BY libelle_mat, nom_etud

4. SELECT code_mat, avg(coeff_mat*note)
   FROM matiere, evaluer
   WHERE matiere.code_mat=evaluer.code_mat
   GROUP BY code_mat

5. SELECT N°etud , avg(coeff_mat*note)
   FROM etudiant , matiere , evaluer
   WHERE etudiant.N°etud=evaluer.N°etud
   AND matiere.code_mat=evaluer.code_mat
   GROUP BY N°etud

6. SELECT avg(coeff_mat*note)
   FROM matiere , evaluer
   WHERE matiere.code_mat = evaluer.code_mat

7. SELECT nom_etud , avg(coeff_mat*note)
   FROM etudiant , matiere , evaluer
   WHERE etudiant.N°etud=evaluer.N°etud
   AND matiere.code_mat=evaluer.code_mat
   GROUP BY nom_etud
   HAVING avg(coeff_mat*note)>(SELECT avg(coeff_mat*note)
   FROM matiere, evaluer
   WHERE matiere.code_mat=evaluer.code_mat)

Sol. exo 10

1. SELECT count(N°pilote)
   FROM pilote
   WHERE adresse_pil=’Béchar’

2. SELECT SUM(nbre_places)
   FROM avion

3. SELECT N°vol
   FROM vol
   WHERE N°pilote IN (SELECT N°pilote
   FROM pilote
   WHERE adresse_pil=’Béchar’)

4. SELECT nom_pil
   FROM pilote, vol
   WHERE pilote.N°pilote=vol.N°pilote
   AND N°avion=all(SELECT N°avion
   FROM avion)