```sql
/**
  Indicateur 1
  Requête permettant de connaitre le nombre d'utilisateurs total
  Depuis la création du site
 */
SELECT COUNT(*) AS utilisateur
FROM UTILISATEUR;

/**
  Indicateur 2
  Requête permettant de connaitre le nombre d'utilisateur particulier total
  Depuis la création du site
 */
SELECT COUNT(*) as Particulier
FROM UTILISATEURPARTICULIER;

/**
  Indicateur 3
  Requête permettant de connaitre le nombre d'associations total
  Depuis la création du site
 */
SELECT COUNT(*) AS associations
FROM UTILISATEURASSOCIATION;

/**
  Indicateur 4
  Requête permettant de connaitre le nombre d'entrepreneur total
  Depuis la création du site
 */
SELECT COUNT(*) AS entrepreneur
FROM UTILISATEURENTREPRENEUR;

/**
  Indicateur 5
  Requête permettant de connaitre le nombre d'entreprises total
  Depuis la création du site
 */
SELECT COUNT(*) AS entreprises
FROM UTILISATEURENTREPRISE;

/**
  Indicateur 6
  Requête permettant de connaitre le nombre de demande total effectué
  Depuis la création du site
 */
SELECT COUNT(*) AS demande
FROM DEMANDE;

/**
  indicateur 7
  Requête permettant de connaitre le nombre de catégories disponibles sur
le site
 */
SELECT COUNT(*) AS categories
FROM CATEGORIE;

/**
  Indicateur 8
  Requête permettant de connaitre le nombre de service total proposés
  Depuis la création du site
 */
```

```sql
SELECT COUNT(*) as Service
FROM SERVICES;

/**
  Indicateur 9
  Requête permettant de connaitre le nombre de bien total proposés
  Depuis la création du site
 */
SELECT COUNT(*) as Bien
FROM BIEN;

/**
  Indicateur 10
  Requête permettant de connaitre le nombre de publicité total publiés
  Depuis la création du site
 */
SELECT COUNT(*) as PUBLICITE
FROM PUBLICITE;

/**
  Indicateur 11
  Requête permettant de connaitre le nombre d'utilisateurs étants abonés au
news
  Depuis la création du site
 */
SELECT COUNT(*) as abonnements
FROM UTILISATEUR
WHERE ABONNEMENT = 1;

/**
  Indicateur 12
  Requête permettant de connaitre le nombre total de vendeurs
  Depuis la création du site
 */
SELECT COUNT(*) AS vendeur
FROM VENDEUR;

/**
  Indicateur 13
  Requête permettant de connaitre le nombre total de clients
  Depuis la création du site
 */
SELECT COUNT(*) AS clients
FROM CLIENTS;

/**
  Indicateur 14
  Requête permettant de connaitre le nombre d'avis créés depuis la création
du site
  Depuis la création du site
 */
SELECT COUNT(*) AS avis
FROM AVIS;


/**
  Indicateur 15
  Requête permettant de connaitre pour tous les utilisateurs le nombres
d'avis recus
  Depuis la création du site
 */
```

```sql
select idUtilisateur, nbrAvisTotal
from utilisateur
group by idUtilisateur, nbrAvisTotal;


/**
  Indicateur 16
  Requête permettant de connaître le nombre d'utilisateurs pour chaque
ville
 */
select count(*) as utilisateurs, UTILISATEUR.CODEPOSTAL
from utilisateur
group by CODEPOSTAL;


/**
  Indicateur 17
  Requête permettant d'afficher le nombre d'utilisateurs entrepreneurs
classé par metier
 */
select count(*) as utilisateurs_Entrepreneurs, metier
from utilisateurentrepreneur
group by metier;


/**
  Indicateur 18
  View qui permet de voir le nombres de demandes par code postaux
 */
CREATE OR REPLACE VIEW demande_par_zone_geographique AS
SELECT CODEPOSTAL AS codeP, COUNT(*) AS total
FROM DEMANDE
         INNER JOIN UTILISATEUR U on U.IDUTILISATEUR =
DEMANDE.IDUTILISATEUR
GROUP BY U.CODEPOSTAL;

SELECT *
FROM demande_par_zone_geographique;

/**
  Indicateur 19
  Procédure permettant d'afficher la moyenne des notes en fonction de
l'input de l'utilistaeur
  @param : input -> type NUMBER
  Le numéro rentré en paramère correspond aux données auxquelles
l'utilisateur souhaite accéder
 */
CREATE OR REPLACE PROCEDURE affiche_moyenne_notes(input NUMBER)
AS
    badValue exception;
    total        FLOAT;
    numerateur   FLOAT;
    denominateur FLOAT;
BEGIN
    CASE TO_NUMBER(input)
        WHEN 1 THEN SELECT SUM(UTILISATEUR.MOYENNENOTES) into numerateur
FROM UTILISATEUR; -- choix 1
                    SELECT COUNT(*) into denominateur FROM UTILISATEUR;
                    total := numerateur / denominateur;
                    IF total < 1 THEN
                        dbms_output.PUT_LINE('La moyenne des notes de tous
```

```
les utilisateurs est de : 0' || total);
                    ELSE
                        dbms_output.PUT_LINE('La moyenne des notes de tous
les utilisateurs est de : ' || total);
                    end if;
        WHEN 2 THEN SELECT SUM(UTILISATEUR.MOYENNENOTES) -- choix 2
                    into numerateur
                    FROM UTILISATEUR
                             INNER JOIN UTILISATEURENTREPRISE
                                        ON UTILISATEUR.IDUTILISATEUR =
UTILISATEURENTREPRISE.IDUTILISATEURENTREPRISE;
                    SELECT COUNT(*) into denominateur FROM
UTILISATEURENTREPRISE;
                    total := numerateur / denominateur;
                    IF total < 1 THEN
                        dbms_output.PUT_LINE('La moyenne des notes de
toutes les entreprises est de : 0' || total);
                    ELSE
                        dbms_output.PUT_LINE('La moyenne des notes de
toutes les entreprises est de : ' || total);
                    end if;
        WHEN 3 THEN SELECT SUM(UTILISATEUR.MOYENNENOTES) -- choix 3
                    into numerateur
                    FROM UTILISATEUR
                             INNER JOIN UTILISATEURASSOCIATION U
                                        on UTILISATEUR.IDUTILISATEUR =
U.IDUTILISATEURASSOCIATION;
                    SELECT COUNT(*) into denominateur FROM
UTILISATEURASSOCIATION;
                    total := numerateur / denominateur;
                    IF total < 1 THEN
                        dbms_output.PUT_LINE('La moyenne des notes de
toutes les associations est de : 0' || total);
                    ELSE
                        dbms_output.PUT_LINE('La moyenne des notes de
toutes les associations est de : ' || total);
                    end if;
        WHEN 4 THEN SELECT SUM(UTILISATEUR.MOYENNENOTES) -- choix 4
                    into numerateur
                    FROM UTILISATEUR
                             INNER JOIN UTILISATEURENTREPRENEUR U2
                                        on UTILISATEUR.IDUTILISATEUR =
U2.IDUTILISATEURENTREPRENEUR;
                    SELECT COUNT(*) into denominateur FROM
UTILISATEURENTREPRENEUR;
                    total := numerateur / denominateur;
                    IF total < 1 THEN
                        dbms_output.PUT_LINE('La moyenne des notes de tous
les entrepreneurs est de : 0' || total);
                    ELSE
                        dbms_output.PUT_LINE('La moyenne des notes de tous
les entrepreneurs est de : ' || total);
                    end if;
        WHEN 5 THEN SELECT SUM(UTILISATEUR.MOYENNENOTES) -- choix 5
                    into numerateur
                    FROM UTILISATEUR
                             INNER JOIN UTILISATEURPARTICULIER U3
                                        on UTILISATEUR.IDUTILISATEUR =
U3.IDUTILISATEURPARTICULIER;
                    SELECT COUNT(*) into denominateur FROM
UTILISATEURPARTICULIER;
```

```
                    total := numerateur / denominateur;
                    IF total < 1 THEN
                        dbms_output.PUT_LINE('La moyenne des notes de tous
les particuliers est de : 0' || total);
                    ELSE
                        dbms_output.PUT_LINE('La moyenne des notes de tous
les particuliers est de : ' || total);
                    end if;
        ELSE RAISE badValue; -- Si l'input n'est pas compris entre 1 et 5
ou alors que ce n'est pas un nombre
        END CASE;
EXCEPTION
    WHEN badValue THEN dbms_output.PUT_LINE('Le numéro donné ne correspond
pas !');
    WHEN others THEN dbms_output.PUT_LINE('Erreur !');
end;
/

BEGIN
    /**
      Si vous voulez la moyenne de tous les utilisateurs entrez (1)
      Si vous voulez la moyenne de toutes les entreprises entrez (2)
      Si vous voulez la moyenne de toutes les associations entrez (3)
      Si vous voulez la moyenne de tous les entrepreneurs entrez (4)
      Si vous voulez la moyenne de tous les particuliers entrez (5)
     */
    affiche_moyenne_notes(1);
end;
/


/**
  Indicateur 20
  Procédure permettant de savoir le nombres d'utilisateurs se situants dans
le code postal rentré en paramètre
  @Param : codeP -> même type que l'attribut code postal se situant dans la
talbe Localisation
 */
CREATE OR REPLACE PROCEDURE affiche_nbr_pour_region(codeP
LOCALISATION.CODEPOSTAL%type)
AS
    ex_code_invalide exception;
    ex_code_trop_long exception;
    pragma exception_init ( ex_code_trop_long, -6502 ); -- Si code postal
trop long raise exception
    total INTEGER;
BEGIN
    IF codeP = TO_NUMBER(codeP) AND LENGTH(codeP) = 5 THEN -- Si le
code_postal a bien une longueur de 5 et qu'il est
    -- constitué d'entiers alors on calcul
        select count(*) into total from localisation where codePostal like
codeP;
        DBMS_OUTPUT.PUT_LINE('Il y a ' || total || ' utilisateurs se
situant dans la région : ' || codeP);
    ELSE
        RAISE ex_code_invalide; -- le code_postal ne correspond pas
    END IF;
EXCEPTION
    WHEN ex_code_trop_long THEN dbms_output.put_line('Code postal trop long
!');
    WHEN ex_code_invalide THEN dbms_output.put_line('Code postal invalide
```

```
!');
    WHEN others THEN dbms_output.put_line('Erreur !');
END;
/

BEGIN
    affiche_nbr_pour_region('57340');
end;
/


/**
  Indicateur 21
  Procédure affichant la ou les categories les moins utilisées
 */
CREATE OR REPLACE PROCEDURE categorie_moins_utilises
    IS
    temp             integer;
    temp_bien        integer;
    temp_service     integer;
    min_categorie    integer := 9999;
    nom_categorie    SERVICES.nom%TYPE;
    number_categorie integer;
BEGIN
    SELECT COUNT(*) INTO number_categorie FROM CATEGORIE;
    for i in 1..number_categorie -- première boucle pour trouver le minimum
        LOOP
            temp := 0;
            SELECT COUNT(*) into temp_bien FROM BIEN WHERE IDCATEGORIE = i;
            SELECT COUNT(*) into temp_service FROM SERVICES WHERE
IDCATEGORIE = i;
            temp := temp_bien + temp_service;
            IF temp < min_categorie THEN
                min_categorie := temp;
            end if;
        end loop;

    for i in 1..number_categorie -- deuxième boucle pour afficher toutes
les catégories les moins utilisés
        LOOP
            temp := 0;
            SELECT COUNT(*) into temp_bien FROM BIEN WHERE IDCATEGORIE = i;
            SELECT COUNT(*) into temp_service FROM SERVICES WHERE
IDCATEGORIE = i;
            temp := temp_bien + temp_service;
            IF temp = min_categorie THEN
                SELECT CATEGORIE.NOM INTO nom_categorie FROM CATEGORIE
WHERE IDCATEGORIE = i;
                DBMS_OUTPUT.PUT_LINE('La catégorie la moins utilisée est :
' || nom_categorie);
            end if;
        end loop;
end;
/

BEGIN
    categorie_moins_utilises();
end;
/
```

```
/**
  Indicateur 22
  Procédure affichant la ou les categories les plus utilisées
 */
CREATE OR REPLACE PROCEDURE categorie_plus_utilises
    IS
    temp             integer;
    temp_bien        integer;
    temp_service     integer;
    max_categorie    integer := 0;
    nom_categorie    SERVICES.nom%TYPE;
    number_categorie integer;
BEGIN
    SELECT COUNT(*) INTO number_categorie FROM CATEGORIE;
    for i in 1..number_categorie -- première boucle pour trouver le maximum
        LOOP
            temp := 0;
            SELECT COUNT(*) into temp_bien FROM BIEN WHERE IDCATEGORIE = i;
            SELECT COUNT(*) into temp_service FROM SERVICES WHERE
IDCATEGORIE = i;
            temp := temp_bien + temp_service;
            IF temp > max_categorie THEN
                max_categorie := temp;
            end if;
        end loop;

    for i in 1..number_categorie -- deuxième boucle pour afficher toutes
les catégories les plus utilisés
        LOOP
            temp := 0;
            SELECT COUNT(*) into temp_bien FROM BIEN WHERE IDCATEGORIE = i;
            SELECT COUNT(*) into temp_service FROM SERVICES WHERE
IDCATEGORIE = i;
            temp := temp_bien + temp_service;
            IF temp = max_categorie THEN
                SELECT CATEGORIE.NOM INTO nom_categorie FROM CATEGORIE
WHERE IDCATEGORIE = i;
                DBMS_OUTPUT.PUT_LINE('La/Les catégories la/les plus
utilisée(s) est/sont : ' || nom_categorie);
            end if;
        end loop;
end;
/

BEGIN
    categorie_plus_utilises();
end;
/

/**
  Indicateur 23
  Procédure affichant le ou les utilisateurs ayants la moins bonnes
notations
 */
CREATE OR REPLACE PROCEDURE utilisateurs_moins_bien_notes
    IS
    CURSOR c$id_u is SELECT IDUTILISATEUR, SUM(AVIS.NOTEUTILISATEUR) AS
somme, COUNT(IDUTILISATEUR) AS total
                     FROM AVIS
                     GROUP BY IDUTILISATEUR
                     ORDER BY IDUTILISATEUR;
```

```
    c$avis          c$id_u%ROWTYPE;
    temp            integer;
    min_utilisateur integer := 9999;
BEGIN
    OPEN c$id_u; -- première boucle pour trouver le maximum
    FETCH c$id_u INTO c$avis;
    WHILE c$id_u%FOUND
        LOOP
            temp := 0;
            temp := c$avis.somme / c$avis.total;
            IF temp < min_utilisateur THEN
                min_utilisateur := temp;
            end if;
            FETCH c$id_u INTO c$avis;
        end loop;
    CLOSE c$id_u;

    OPEN c$id_u; -- première boucle pour trouver le maximum
    FETCH c$id_u INTO c$avis;
    WHILE c$id_u%FOUND
        LOOP
            temp := 0;
            temp := c$avis.somme / c$avis.total;
            IF temp = min_utilisateur THEN
                DBMS_OUTPUT.PUT_LINE('L"id ou les id utilisateur(s) ayant
la/les meilleurs notation(s) sont : ' ||
                                     c$avis.IDUTILISATEUR);
            end if;
            FETCH c$id_u INTO c$avis;
        end loop;
    CLOSE c$id_u;
end;
/

BEGIN
    utilisateurs_moins_bien_notes();
end;
/

/**
  Indicateur 24
  Procédure affichant le ou les utilisateurs ayants les meilleurs notations
 */
CREATE OR REPLACE PROCEDURE utilisateurs_mieux_notes
    IS
    CURSOR c$id_u is SELECT IDUTILISATEUR, SUM(AVIS.NOTEUTILISATEUR) AS
somme, COUNT(IDUTILISATEUR) AS total
                     FROM AVIS
                     GROUP BY IDUTILISATEUR
                     ORDER BY IDUTILISATEUR;
    c$avis          c$id_u%ROWTYPE;
    temp            integer;
    min_utilisateur integer := -1;
BEGIN
    OPEN c$id_u; -- première boucle pour trouver le maximum
    FETCH c$id_u INTO c$avis;
    WHILE c$id_u%FOUND
        LOOP
            temp := 0;
            temp := c$avis.somme / c$avis.total;
            IF temp > min_utilisateur THEN
```

```
                min_utilisateur := temp;
            end if;
            FETCH c$id_u INTO c$avis;
        end loop;
    CLOSE c$id_u;

    OPEN c$id_u; -- première boucle pour trouver le maximum
    FETCH c$id_u INTO c$avis;
    WHILE c$id_u%FOUND
        LOOP
            temp := 0;
            temp := c$avis.somme / c$avis.total;
            IF temp = min_utilisateur THEN
                DBMS_OUTPUT.PUT_LINE('L"id ou les id utilisateur(s) ayant
la/les meilleurs notation(s) sont : ' ||
                                     c$avis.IDUTILISATEUR);
            end if;
            FETCH c$id_u INTO c$avis;
        end loop;
    CLOSE c$id_u;
end;
/

BEGIN
    utilisateurs_mieux_notes();
end;
/


/**
  Indicateur 25
  Procedure permettant d'afficher le nombres de ventes de biens en fonction
du code postal
  @Param : code_postal -> même type que l'attribut code postal se situant
dans la table Localisation
 */
create or replace procedure affiche_biens_localisation(code_postal
LOCALISATION.CODEPOSTAL%type)
    is
    id_localisation int;
    nb_ventes       int;
begin
    select idlocalisation into id_localisation from localisation where
code_postal = codepostal;
    select count(*) into nb_ventes from bien where idlocalisation =
id_localisation;
    dbms_output.put_line('Il y a ' || nb_ventes || ' vente(s).');
EXCEPTION
    WHEN no_data_found THEN DBMS_OUTPUT.PUT_LINE('Code postal inexistant
!');
    WHEN others THEN DBMS_OUTPUT.PUT_LINE('Erreur !');
end;
/

BEGIN
    affiche_biens_localisation('57340');
end;
/

/**
  Indicateur 26
```

```
  Procedure permettant d'afficher le nombres de propositions de services en
fonction du code postal
  @Param : code_postal -> même type que l'attribut code postal se situant
dans la table Localisation
 */
create or replace procedure affiche_services_localisation(code_postal
LOCALISATION.CODEPOSTAL%type)
    is
    id_localisation int;
    nb_ventes       int;
begin
    select idlocalisation into id_localisation from localisation where
code_postal = codepostal;
    select count(*) into nb_ventes from services where idlocalisation =
id_localisation;
    dbms_output.put_line('Il y a ' || nb_ventes || ' proposition(s) de
services.');
EXCEPTION
    WHEN no_data_found THEN DBMS_OUTPUT.PUT_LINE('Code postal inexistant
!');
    WHEN others THEN DBMS_OUTPUT.PUT_LINE('Erreur !');
end;
/

BEGIN
    affiche_services_localisation('57340');
end;
/

/**
  Indicateur 27
  Procédure affichant le nombre de ventes pour une date donnée
  @Param : dateh -> même type que l'attribut date_heure se situant dans la
table Planning
 */
create or replace procedure affiche_total_vente_pour_ville(dateh
planning.date_heure%type)
    is
    nb_ventes int;
begin
    select count(*) into nb_ventes from planning where dateh = date_heure;
    dbms_output.put_line('Il y a ' || nb_ventes || ' de vente le ' ||
dateh);
EXCEPTION
    WHEN no_data_found THEN DBMS_OUTPUT.PUT_LINE('Date inexistante !');
    WHEN others THEN DBMS_OUTPUT.PUT_LINE('Erreur !');
end;
/

BEGIN
    affiche_total_vente_pour_ville(TO_DATE('27/05/22', 'dd/mm/yy'));
end;
/


/**
  Indicateur 28
  Procédure affichant toutes les ventes créés/effectuées pour toutes les
dates disponibles sur le site
 */
create or replace procedure affiche_vente_tout_planning
```

```
    is
    nb_ventes int;
    dateh     planning.date_heure%type;
begin
    for i in (select * from bien)
        loop
            select count(*) into nb_ventes from bien where i.IDPLANNING =
idplanning;
            select date_heure into dateh from planning where i.IDPLANNING =
idplanning;
            dbms_output.put_line('Il y a ' || nb_ventes || ' vente(s) le '
|| dateh);
        end loop;
end;
/

BEGIN
    affiche_vente_tout_planning();
end;
/

/**
  Indicateur 29
  Procédure affichant tous les services créés/effectuées pour toutes les
dates disponibles sur le site
 */
create or replace procedure affiche_services_tout_planning
    is
    nb_ventes int;
    dateh     planning.date_heure%type;
begin
    for i in (select * from services)
        loop
            select count(*) into nb_ventes from services where i.IDPLANNING
= idplanning;
            select date_heure into dateh from planning where i.IDPLANNING =
idplanning;
            dbms_output.put_line('Il y a ' || nb_ventes || ' service(s)
proposé(s) le ' || dateh);
        end loop;
end;
/

BEGIN
    affiche_services_tout_planning();
end;
/
```