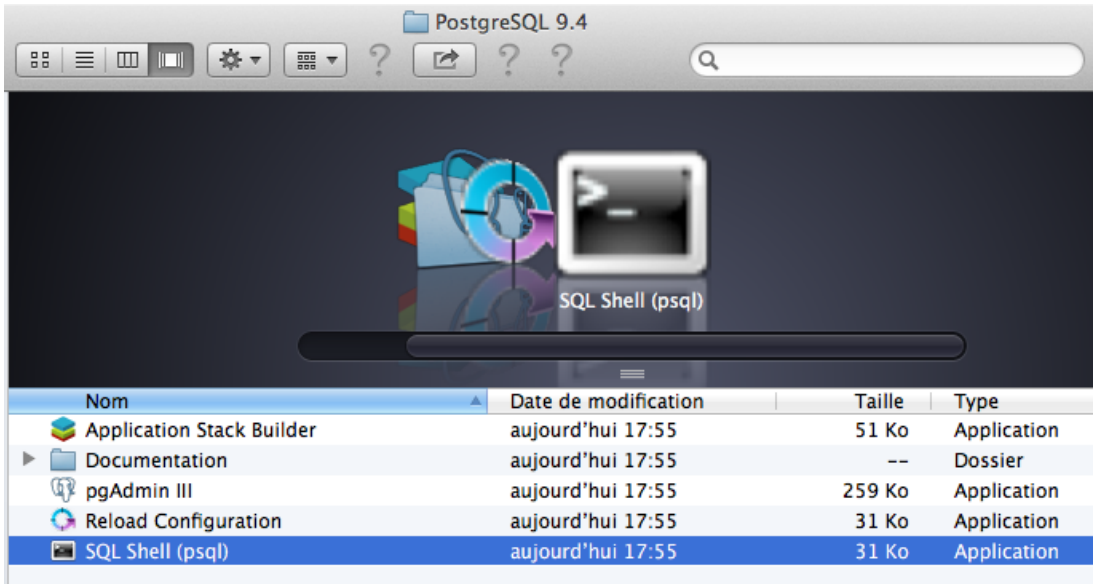
# Bases de données[1]
# TD◦5 - PL/SQL

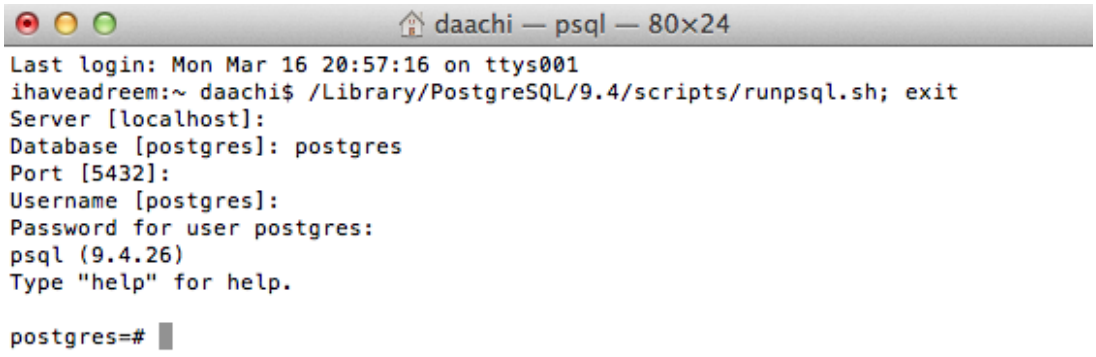
Dans chaque exercice et pour chaque question, le script et la capture d'écran du résultat d'exécution sont exigés

## Installation de PostGress

— Choisir la version `PostGreSQL 9.4`
— Choisir celle qui correspond au système d'exploitation que vous utilisez (Unix, Windows, Mac).
— Lors de son installation, il vous sera demandé un mot de passe. Mettez un mot de passe simple que vous pouvez retenir facilement.
— Après l'installation, vous devriez logiquement avoir un répertoire dont les éléments sont montrés dans la figure suivante :




— Cliquer sur `SQL Shell` et vous devriez avoir quelque chose qui ressemble à la fenêtre de la figure suivante :




Seul le champ `Database` et `password` choisi lors de l'installation devraient être renseignés comme indiqué dans cette figure. Ne rien mettre pour les autres champs.
— Pour créer votre propre base de données, vous y connecter, créer une table et voir ses champs avec leur types, vous pouvez faire comme dans la figure suivante.

1. `Année universitaire 2019-2020.`

```
postgres=# create database base_exemple;
CREATE DATABASE
postgres=# \c base_exemple;
You are now connected to database "base_exemple" as user "postgres".
base_exemple=# CREATE TABLE EMP(
base_exemple(# EMPNO INTEGER NOT NULL,
base_exemple(#         ENAME VARCHAR(15),
base_exemple(#         JOB VARCHAR(14),
base_exemple(#         MGR INTEGER,
base_exemple(#         HIREDATE DATE,
base_exemple(#         SAL REAL,
base_exemple(#         COMM REAL,
base_exemple(#         DEPTNO INTEGER);
CREATE TABLE
base_exemple=# \d emp
             Table "public.emp"
  Column  |         Type          | Modifiers
----------+-----------------------+-----------
 empno    | integer               | not null
 ename    | character varying(15) |
 job      | character varying(14) |
 mgr      | integer               |
 hiredate | date                  |
 sal      | real                  |
 comm     | real                  |
 deptno   | integer               |

base_exemple=#
```

Pour les exercices suivants, on considère les tables `EMP` et `DEPT`[2] :

| NoEMP | NomEMP | EMPLOI | MGR | DateEMB | SAL | COMM | NoDEPT |
|---|---|---|---|---|---|---|---|
| 7369 | SERGE | FONCTIONNAIRE | 7902 | 17-DEC-1980 | 800 | | 20 |
| 7499 | BRAHIM | VENDEUR | 7698 | 20-FEB-1981 | 1600 | 300 | 30 |
| 7521 | NASSIMA | VENDEUR | 7698 | 22-FEB-1981 | 1250 | 500 | 30 |
| 7566 | LUCIE | GESTIONNAIRE | 7839 | 2-APR-1981 | 2975 | | 20 |
| 7654 | MARTIN | VENDEUR | 7698 | 28-SEP-1981 | 1250 | 1400 | 30 |
| 7698 | BENJAMIN | GESTIONNAIRE | 7839 | 1-MAY-1981 | 2850 | | 30 |
| 7782 | DAYANE | GESTIONNAIRE | 7839 | 9-JUN-1981 | 2450 | | 10 |
| 7788 | ARIJ | ANALYSTE | 7566 | 09-DEC-1982 | 3000 | | 20 |
| 7839 | MAYAR | PRESIDENT | | 17-NOV-1981 | 5000 | | 10 |
| 7844 | ROI | VENDEUR | 7698 | 8-SEP-1981 | 1500 | 0 | 30 |
| 7876 | VIRGINIE | FONCTIONNAIRE | 7788 | 12-JAN-1983 | 1100 | | 20 |
| 7900 | LYNA | FONCTIONNAIRE | 7698 | 3-DEC-1981 | 950 | | 30 |
| 7902 | ASMA | ANALYSTE | 7566 | 3-DEC-1981 | 3000 | | 20 |
| 7934 | SIMONE | FONCTIONNAIRE | 7782 | 23-JAN-1982 | 1300 | | 10 |

| NoDEPT | NomDEPT | LOC |
|---|---|---|
| 10 | COMPTABILITE | BREST |
| 20 | RECHERCHE | RENNES |
| 30 | VENTES | DINAR |
| 40 | GESTION | DINAN |

**Exercice 1** *Blocs PL/SQL anonymes*

1. Écrire un bloc `PL/SQL` calculant le maximum de deux nombres et le renvoie en sortie
2. Écrire un bloc `PL/SQL` qui affiche le nombre d'employés d'un département dont le numéro est donné.
3. Créer une table `NOMBRE` ayant les deux attributs de type `NUMBER, A` et `B`.
   — `B` est fonction de `A`
   — Ecrire un bloc `PL/SQL` qui remplit cette table comme suit :

2. `EMP pour Employé` et `DEPT pour Département`

| A | B |
|---|---|
| 1 | 3 |
| 2 | 6 |
| 3 | 9 |
| 4 | 12 |
| 5 | 15 |
| ... | ... |
| 10 | 30 |

4. Écrire un programme PL/SQL affichant le reste de la division de `n` par `m` sans utiliser la fonction `modulo`.
   — Les nombres `n` et `m` sont deux entiers strictement positifs.
   — Traiter le cas ou `m` vaut `0`.
5. On souhaite afficher pour un département dont le numéro est donné, le message :

   `"Le département NOM se trouve à LOC".`

   Si par exemple le numéro donné est 30, le message sera :

   `"Le département VENTES se trouve à DINAR"`

   Proposer un programme PL/SQL pour assurer cette opération.

**Exercice 2** *Procédures et fonctions*

1. Écrire une fonction PL/SQL qui calcule et affiche `n!`.
   — `n` est un entier et doit être considéré comme paramètre de la fonction
   — Proposer une solution itérative et une solution récursive.
2. Écrire un programme PL/SQL qui contient les éléments suivants :
   — Une fonction récursive `fFactR (n in number)` calculant `n!`.
   — Une procédure `pFact (n in number)` qui appelle la fonction `fFactR`.
   — Un bloc PL/SQL anonyme qui appelle la procédure `pFact`.
   — `n` doit être lu en dehors de `fFactR` et de `pFact`.
3. Proposer une procédure affichant les nombres de `1` à `n`.
   — `n` est un entier et doit être considéré comme paramètre de la procédure
   — Proposer deux façons de tester le fonctionnement de cette procédure.
4. Proposer une fonction qui calcule et renvoie $2n + m^2$.
   — `n et m` sont des entiers et doivent être considérés comme paramètres de la fonction.
   — Aucune autre variable ne doit être déclarée.
   — Proposer deux façons de tester le fonctionnement de cette fonction.
5. Écrire un programme PL/SQL qui contient les éléments suivants :
   — Une procédure `pAug(nEmp number, tAug number)` augmentant de `tAug%` le salaire de l'employé identifié par `nEmp`.
   — Un bloc PL/SQL anonyme qui appelle la procédure `pAug`.
   — `nEmp` et `tAug` doivent être lus en dehors de `pAug`.

**Exercice 3** *Utilisation de curseurs*

1. Écrire un bloc PL/SQL affichant le nombre de départements dont aucun employé ne touche une commission. Un département possédant un ou plusieurs employés ayant une commission `>0` ne doit pas être considéré.
2. Écrire un bloc PL/SQL affichant le nombre de départements dont tous les employés touchent un salaire supérieur à 2000€.
3. Proposer un bloc PL/SQL qui affiche les noms des départements qui existent dans les deux tables `EMP` et `DEPT`.
4. Proposer un bloc PL/SQL qui affiche le résultat suivant :

| NDEPT | NOMBEMP | MS1 | MS2 |
|---|---|---|---|
| | | | |

   Avec :
   — `NDEPT` : nom du département
   — `NOMBEMP` : nombre d'employés
   — `MS1` est la moyenne des salaires par département. Seuls les salaires inférieurs ou égaux à `2500€` sont à considérer.
   — `MS2` est la moyenne des salaires par département. Seuls les salaires supérieurs à `2500€` sont à considérer.
   — Seuls les employés n'ayant pas de commission sont à considérer.
5. Proposer un bloc PL/SQL affichant les noms des employés du département `"VENTES"` dont le salaire `> 1400 €`.
6. Proposer un bloc PL/SQL permettant de recopier le contenu de la table `EMP` avec augmentation de la commission de `50%` dans la table `EMPBIS` possédant le même schéma que `EMP`. On suppose que la table `EMPBIS` existe et est vide.

7. Proposer un bloc `PL/SQL` qui vérifie à partir de son nom donné, l'existence d'un département (par son numéro) dans la table `EMP`.

**Exercice 4** *Déclencheurs/Triggers*

1. On souhaite interdire toute opération n'ayant pas pour objet la simple consultation de la table `DEPT` après `17H00`. Proposer un déclencheur assurant cette tâche.
2. On souhaite interdire de manière automatique toute insertion d'un salaire `< 500€`. Proposer un déclencheur assurant cette tâche.