SQL (postgresql)

```
-- [...] = optionnel

CREATE TYPE TYPE_DEPENSE AS ENUM ('materiel', 'fonctionnement');

DROP TABLE IF EXISTS MATABLE CASCADE; -- CASCADE supprime les vues qui en dépendent, et les contraintes FK dans les autres tables
CREATE TABLE MEMBRE_LABO (
  matable_id SERIAL PRIMARY KEY, -- SERIAL = INTEGER + crée une séquence automatiquement ; PRIMARY KEY = UNIQUE + NOT NULL
  categorie TEXT NOT NULL CHECK (type = 'I' OR type = 'B'),
  mail_provider FLOAT [CONSTRAINT nom_de_la_contrainte] UNIQUE, -- NULL est autorisé plusieurs fois
  telephone VARCHAR(22),
  date_debut DATE, -- 'YYYY-MM-DD'
  rdv TIMESTAMP REFERENCES RENDEZVOUS, -- 'YYYY-MM-DD HH:MM:SS' -- Référence la primary key de la table RENDEZVOUS
  statut BOOLEAN,
  depense TYPE_DEPENSE NOT NULL
  CHECK (date_debut IS NOT NULL),
  CHECK (mail_provider BETWEEN 1.2 AND 1.5),
  UNIQUE (date_debut, rdv), -- On peut aussi mettre PRIMARY KEY ici, si on ne l'a pas écrit plus haut
  FOREIGN KEY(mail_provider,telephone) REFERENCES USER(mail,telephone) ON DELETE CASCADE -- Les entrées seront supprimées quand le
      tuple (mail,telephone) n'existe plus dans USER
);

CREATE SEQUENCE serial [INCREMENT BY 2] [MINVALUE 1] [MAXVALUE 500] [START 101] [CYCLE]; -- CYCLE = on recommence quand on atteint
    MAXVALUE
SELECT currval('serial'); -- On peut faire un INSERT INTO xxx VALUES (nextval('serial'));

CREATE VIEW VINGENIEUR AS SELECT id,nom,mail,telephone,validateur,labo,specialite FROM MEMBRE_LABO WHERE type = 'I';

CREATE FUNCTION CHECK_DATE_PROJET_CREATE() RETURNS trigger AS $ret$
  DECLARE
    proposition date;
  BEGIN
    SELECT INTO proposition date_emission FROM PROPOSITION_DE_PROJET prop WHERE prop.appel = NEW.appel;
    IF(NEW.date_debut >= proposition) THEN
      RETURN NEW; -- Les procédure appelées par un trigger BEFORE doivent retourner NEW en cas de succès
    ELSE
      RAISE EXCEPTION 'Date de début antérieure à la proposition de projet !';
  END IF;
  RETURN NULL;
  END;
$ret$ LANGUAGE 'plpgsql'; -- Obligatoire

CREATE TRIGGER TRIGGER_CHECK_DATE_PROJET_CREATE
  BEFORE INSERT ON PROJET
  FOR EACH ROW EXECUTE PROCEDURE CHECK_DATE_PROJET_CREATE();
-------------------------------------------------------------------------------------------------------------------------------------------
BEGIN [TRANSACTION]; -- Début transaction
SELECT [DISTINCT] * | colonne [, ...] FROM unetable [, ... ] WHERE condition GROUP BY expression HAVING condition ORDER BY expression
      [ ASC | DESC ] LIMIT nombre; -- Possibilité de mettre un alias pour colonne (avec le mot-clé AS) et un alias pour table
-- colonne peut contenir COUNT(*), SUM(nb*prix), etc. HAVING condition peut contenir COUNT(id) > 5 par exemple.

INSERT INTO table [(colonne [, ...])] VALUES ({expression | DEFAULT} [, ...]) [RETURNING * | expression [[AS] nomdesortie] [, ...]];

UPDATE table SET {colonne = {expression | DEFAULT}} [, ...] WHERE condition [RETURNING * | expression [[AS] nomdesortie] [, ...]];

DELETE FROM table WHERE condition [RETURNING * | expression [[AS] nomdesortie] [, ...]];
END; -- Fin transaction, COMMIT implicite
```

PHP et PDO

```php
$dbh = new PDO("pgsql:host=domain.tld;port=5432;dbname=database", "username", "password"); // Remplacer domain.tld, database,
    username et password, port est facultatif, déclenche une PDOException en cas d'erreur lors de la connexion
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
//-----------------------------------------------------------------------------------------------------------------------------
$res = $this->db->query("SELECT * from MATABLE"); // resultat de la requete contenu dans $res
$rows = $res->fetchAll(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, CLASSE_PHP); // CLASSE_PHP est une classe PHP
// fetchAll est à FETCH_BOTH par défaut (tableau indexé par nom des colonnes et numéro des colonnes, en commençant par 0)
// $rows contient un tableau d'objet de la classe CLASSE_PHP
//-----------------------------------------------------------------------------------------------------------------------------
$sql = "SELECT * FROM MATABLE WHERE num = :numero AND app = :appel";
$params = array(':numero' =>$numero, ':appel' =>$appel);
$query = $dbh->prepare($sql);
$res = $query->execute($params);
if ($res===True && query->rowCount()===1) {
  $row = $query->fetch(PDO::FETCH_OBJ); // Crée un objet anonyme ayant les noms de colonnes comme propriétés
  echo $row->NAME;
}
$dbh = NULL; // Coupe la connexion
```

PHP et pg_connect

```php
$conn = pg_connect("host=domain.tld port=5432 dbname=database user=username password=password") or die("Connexion impossible");
if (!$dbh) {
  echo "Erreur";
  exit;
}
$result = pg_query($conn, "SELECT auteur, email FROM auteurs"); // On peut mettre plusieurs statements séparés par des ";"
if (!$result) {
  echo "Erreur";
  exit;
}
$nombre_de_resultats = pg_num_rows($result);
while ($row = pg_fetch_row($result)) {
  echo "Auteur: $row[0] E-mail: $row[1]";
  echo "<br />\n";
}
//-----------------------------------------------------------------------------------------------------------------------------
$arr = pg_fetch_array($result, 0, PGSQL_BOTH);
echo $arr[0] . " <- Row 1 Author\n";
pg_close($conn);
```