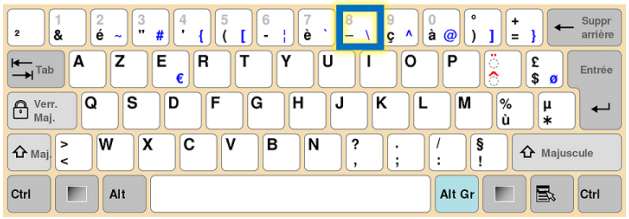
TP3 : DOCUMENT C (APRES SEANCE DE TP)

Grace à l'opérateur LIKE, il est possible de faire des recherches sur des chaines de caractères incomplets en utilisant deux caractères qui sont :

- Le pourcentage « **%** » : il permet de remplacer zéro, un, ou plusieurs caractères.
- L'underscore « la touche 8 du clavier : _ » : il permet de remplacer un seul et unique caractère :



Dans ce tableau différents exemple d'expressions utilisant LIKE (colonne de gauche), ainsi que la valeur renvoyée par l'expression en question (colonne de droite).

Expression	Résultat
'LHADDAD' LIKE '%LHADDAD%'	TRUE
'LHADDAD' LIKE 'LHADDA%'	TRUE
'LHADDAD' LIKE '%AD%'	TRUE
'LHADDAD' LIKE '%HA%AD'	TRUE
'LHADDAD' LIKE '%H%DA%'	TRUE
'LHADDAD' LIKE 'LHAD_AD'	TRUE
'LHADDAD' LIKE 'LHAD__AD'	FALSE
'LHADDAD' LIKE '_LHADDAD'	FALSE
'LHADDAD' LIKE 'LH__DA%'	TRUE
'LHADDAD' LIKE '_H%'	TRUE

Si vous souhaitez tester ces expressions, écrivez dans votre SGBDR (volet SQL):

```
SELECT  'LHADDAD' LIKE '%AD%' ;
```

La réponse sera une table d'une ligne et d'une colonne. En fonction de votre SGBDR, la valeur contenue dans cette table peut varier : **TRUE** est par exemple équivalent à 1 en MySQL ou t en Postgresql ; et **FALSE** est équivalent à 0 en MySQL et f en Postgresql.

Majuscules et minuscules

Les SGBDR n'ont pas tous le même comportement vis-à-vis de la distinction entre majuscules et minuscules. Pour savoir le comportement de notre SGBD MySQL face à la différentiation entre majuscules et minuscules, nous testons le résultat renvoyé par cette requête (certains SGBD vont renvoyer TRUE, d'autres FALSE) :

```
SELECT  'LHADDAD' LIKE '%dd%' ;
```

MySQL renvoie vrai pour cette expression (ce qui veut dire que MySQL est **insensible** à la distinction entre majuscules et minuscules. En d'autre terme pour MySQL « d » et « D » c'est la même chose).

Pour d'autre SGBD sensible à la distinction entre majuscules et minuscules, il est conseillé de s'y prendre comme suit :

```
SELECT  LOWER ('LHADDAD') LIKE '%dd%' ;
```

De cette manière, quelle que soit la valeur entre parenthèse de la fonction LOWER, elle sera converti en minuscules (cette fonction **n'est pas utilisée dans MySQL** mais elle vous sera bien utile pour d'autre SGBD).

LIKE et NOT LIKE:

Nous pouvons savoir si deux chaines de caractères soient différentes ou pas en utilisant l'opérateur NOT LIKE. Exemple :

```
SELECT  'LHADDAD NOUREDINNE' NOT LIKE 'LHAFFAF%' ;
```
Cette requête retourne VRAI car LHADDAD et LHAFFAF sont des chaines de caractères différentes.

```
SELECT  'LHADDAD NOUREDINNE' NOT LIKE '%NOUREDINNE' ;
```
Cette requête retourne FAUX car NOUREDINNE et NOUREDINNE sont des chaines de caractères similaires.

Revenons à l'exercice du TP3 :

Cinéma (**numC**, nomCinéma, numéro, rue, ville)
Salle (**NumSalle**, #numCSal, capacité, climatisée)
Film (**idFilm**, titre, année, genre, résumé)
Séance **(#idFilmS, #numCS, #NumSalleS**, tarif)

Créer la requête SQL permettant de répondre à chacune des questions suivantes :

- Donner les Cinéma dont leurs villes commencent par 'A'.

```
SELECT numC, nomCinéma
FROM Cinéma
WHERE ville LIKE 'A%' ;
```

Test :

```
CREATE TABLE  Cinéma  (
                        numC  INT NOT NULL,
                        nomCinéma VARCHAR(30) NOT NULL,
                        numéro INT NOT NULL,
                        rue VARCHAR(40) NOT NULL,
                        ville VARCHAR(20) NOT NULL,
                        PRIMARY KEY (numC)
                       ) ENGINE=InnoDB DEFAULT CHARSET=utf8 ;
```

```
INSERT INTO Cinéma (numC, nomCinéma, numéro, rue, ville)
VALUES
(1, 'Cinéma L Algeria', 52, 'Didouche Mourad', 'Alger 16000'),
(2, 'Salle Ibn Khaldoun', 12, 'Dr Cherif Saadane', 'Alger  Ctre 16000'),
(3, 'Salle El Mouggar', 01, 'Asselah Hocine', 'Alger Ctre 16000'),
(4, 'Cinématheque d Alger', 49, 'Larbi Ben Mhidi', 'Alger 16000'),
(5, 'Cinéma Little Rex', 01, 'Boulevard Hydra', 'Alger 16000'),
(6, 'Cinema Afrique', 01, 'Rue des Frères Meslem, Sidi MHamed', 'Alger 16000'),
(7, 'Salle Ibn Zeydoun', 01, 'El Madania', 'Alger 16000'),
(8, 'Cinéma Mogador', 01, 'Rue Bahloul Ahmed', 'Oran'),
(9, 'Cinema Mansourah', 01, 'Rue Larbi Ben Mhidi', 'Oran'),
(10, 'Cinéma Le Hoggar (ex Century)', 01, 'Rue Djellaili', 'Oran'),
(11, 'Cinéma Marhaba', 01, 'Bvd Emir AEK', 'Oran'),
(12, 'Cinéma El Feth (ex-Pigalle)', 01, 'Rue S L Hamdani Adda', 'Oran'),
(13, 'Cinema le Méditerranéen', 01, 'Rue Avenue Khiali Ben Salem Mohamed', 'Oran'),
(14, 'Cinéma Zénith Constantine', 01, 'Constantine ville', 'Constantine'),
(15, 'Salle Ahmed Bey', 01, 'Constantine ville', 'Constantine'),
(16, 'Media-Plus', 32, 'Rue Abane Ramdane', ' Constantine 25000'),
(17, 'Cinéma IMAX 7D ', 01,'Constantine ville', 'Constantine'),
 (18, 'Cinémathèque de Bejaia', 01, 'Bejaia ville', 'Ville de Bejaia'),
(19, 'Cinéma Djurdjura', 01, 'Bvd Houari Boumediene', 'Ville de Tizi Ouzou'),
```

(20, 'Cinéma Pictures', 01, 'Rue des Frères Mankour', 'Tizi Ouzou'),
(21, 'Cinéma Mondial', 01, 'Boulevard Houari Boumédiène', 'Oum el Bouaghi');

- Donner les films qui ne contiennent pas le mot « bataille » (la bataille d'Alger).

```
SELECT idFilm, titre
FROM Film
WHERE titre NOT LIKE '%bataille%' ;
```

Test :

```
CREATE TABLE  Film  (
                        idFilm  INT NOT NULL,
                        titre VARCHAR(30) NOT NULL,
                        année YEAR(4) NOT NULL,
                        genre VARCHAR(30) NOT NULL,
                        résumé TEXT,
                        PRIMARY KEY (idFilm)
                       ) ENGINE=InnoDB DEFAULT CHARSET=utf8 ;
```

```
INSERT INTO Film (idFilm, titre, année, genre, résumé)
VALUES
(1, 'La bataille des ardennes', '2007', 'Historique', NULL),
(2, 'The Batman', '2019', 'Action', NULL),
(3, 'La bataille de midway', '1976', 'Historique', NULL),
 (4, 'Alita : Battle Angel', '2018', 'Science-Fiction', NULL),
(5, 'Intouchables', '2020', 'Comédie', 'résumé'),
(6, 'The Host', '2006', 'Fantastique', NULL),
(7, 'LE RETOUR DE CHUCKY', '2017', 'Horreur', NULL),
(8, 'Lord of War', '2006', 'Drame', NULL),
(9, 'Black Storm', '2014', 'Thriller', NULL),
(10, 'Bad Teacher', '2020', 'Comédie-Romantique', NULL),
(11, 'ARMADA La Bataille Navale ', '2018', 'Thriller', NULL),
(12, 'Les Trois Frères', '2020', 'Comédie', NULL),
(13, 'Les Autres', '2001', 'Horreur', NULL) ;
```

- Donner les films dont leurs genres commencent par un caractère suivi par 'or' (horreur).

```
SELECT idFilm, titre
FROM Film
WHERE genre LIKE '_or%' ;
```

Test : Testez à partir des insertions de la question précédente.

- Donner trois fois les tarifs des films dont le nom contient « l ».

Cinéma (**numC**, nomCinéma, numéro, rue, ville)
Salle (**NumSalle**, #numCSal, capacité, climatisée)
Film (**idFilm**, titre, année, genre, résumé)
Séance **(#idFilmS, #numCS, #NumSalleS**, tarif)

```
SELECT Film.idFilm, Film.titre, 3*(Séance.tarif)
FROM Film, Séance
WHERE Film.idFilm=Séance.idFilmS
AND Film.titre LIKE '%l%' ;
```

Test :

```
CREATE TABLE  Salle (
                        NumSalle  INT NOT NULL,
                        numCSal INT NOT NULL,
                        capacité   INT NOT NULL,
                        climatisée  VARCHAR(30) NOT NULL,
                        PRIMARY KEY (NumSalle),
                        FOREIGN KEY (numCSal) REFERENCES  Cinéma (numC)
                        ) ENGINE=InnoDB DEFAULT CHARSET=utf8 ;
```

```
INSERT INTO Salle (NumSalle, numCSal, capacité, climatisée)
VALUES
(1, 1, 150, 'TRUE'),
(2, 1, 170, 'TRUE'),
(3, 1, 200, 'TRUE'),
(4, 1, 100, 'FALSE'),
(5, 1, 100, 'FALSE'),
(6, 1, 300, 'TRUE'),
(7, 1, 400, 'TRUE'),
(8, 1, 400, 'TRUE'),
(9, 1, 400, 'TRUE'),
(10, 1, 150, 'TRUE'),
(11, 1, 120, 'TRUE'),
(12, 1, 120, 'TRUE'),
(13, 1, 135, 'TRUE'),
(14, 1, 190, 'TRUE'),
(15, 1, 150, 'TRUE'),
(16, 1, 150, 'TRUE'),
(17, 2, 150, 'TRUE'),
(18, 2, 170, 'TRUE'),
(19, 2, 200, 'TRUE'),
(20, 2, 100, 'FALSE'),
```

```
(21, 3, 100, 'FALSE'),
(22, 3, 300, 'TRUE'),
(23, 3, 400, 'TRUE'),
(24, 4, 100, 'FALSE'),
(25, 4, 300, 'TRUE'),
(26, 4, 400, 'TRUE'),
(27, 4, 150, 'TRUE'),
(28, 5, 400, 'TRUE'),
(29, 5, 150, 'TRUE'),
(30, 5, 170, 'TRUE'),
(31, 5, 200, 'TRUE'),
 (32, 6, 400, 'TRUE'),
(33, 6, 150, 'TRUE'),
(34, 7, 100, 'FALSE'),
(35, 7, 100, 'FALSE'),
(36, 7, 300, 'TRUE'),
(37, 9, 170, 'FALSE'),
(38, 9, 200, 'TRUE'),
 (39, 9, 400, 'TRUE'),
(40, 10, 150, 'TRUE'),
(41, 10, 100, 'FALSE'),
(42, 10, 100, 'FALSE'),
(43, 10, 300, 'TRUE'),
(44, 14, 170, 'TRUE'),
(45, 14, 200, 'TRUE'),
 (46, 14, 400, 'FALSE'),
(47, 14, 150, 'FALSE'),
(48, 15, 100, 'FALSE'),
(49, 15, 100, 'FALSE'),
(50, 15, 300, 'TRUE');

CREATE TABLE Séance (
                        idFilmS INT NOT NULL,
                        numCS INT NOT NULL,
                        NumSalleS INT NOT NULL,
                        tarif INT NOT NULL,
                       PRIMARY KEY (idFilmS, numCS, NumSalleS),
                       FOREIGN KEY (idFilmS) REFERENCES Film (idFilm),
                       FOREIGN KEY (numCS) REFERENCES Cinéma (numC),
                       FOREIGN KEY (NumSalleS) REFERENCES Salle (NumSalle)
                      ) ENGINE=InnoDB DEFAULT CHARSET=utf8 ;
```

INSERT INTO Séance (idFilmS, numCS, NumSalleS, tarif)
VALUES
(1, 1, 1, 250),
(1, 1, 16, 250),
(1, 2, 17, 250),
(1, 2, 20, 250),
(1, 3, 21, 250),
(1, 3, 23, 250),
(1, 4, 24, 250),
(1, 4, 27, 250),
(1, 5, 28, 250),
(1, 5, 31, 250),
(1, 6, 32, 250),
(1, 6, 33, 250),
(1, 7, 34, 250),
(1, 7, 36, 250),
(1, 9, 37, 250),
(1, 9, 39, 250),
(1, 10, 40, 250),
(1, 10, 43, 250),
(1, 14, 44, 250),
(1, 14, 47, 250),
(1, 15, 48, 250),
(1, 15, 50, 250),

(5, 9, 37, 300),
(6, 9, 38, 350),
(7, 9, 39, 400),
(8, 10, 40, 450),
(9, 10, 41, 500),
(10, 10, 42, 550),
(11, 10, 43, 600);

Opérations Algébriques:

Cinéma (**numC**, nomCinéma, numéro, rue, ville)
Salle (**NumSalle**, #numCSal, capacité, climatisée)
Film (**idFilm**, titre, année, genre, résumé)
Séance **(#idFilmS, #numCS, #NumSalleS**, tarif)

- Donner les cinémas qui se trouvent à Alger « **OU** » qui ont une salle de capacité supérieure à 200 places.

```
SELECT numC, nomCinéma
FROM Cinéma
WHERE ville LIKE '%Alger%'
UNION
SELECT Cinéma.numC, Cinéma.nomCinéma
FROM Cinéma, Salle
WHERE Cinéma.numC= Salle.numCSal
AND  capacité >200 ;
```

On peut également écrire :

```
SELECT numC, nomCinéma
FROM Cinéma
WHERE ville LIKE '%Alger%'
UNION
SELECT numC, nomCinéma
FROM Cinéma
WHERE numC IN  (SELECT numCSal
                FROM Salle
                WHERE capacité >200) ;
```

On peut également écrire :

```
SELECT numC, nomCinéma
FROM Cinéma
WHERE ville LIKE '%Alger%'
UNION
SELECT numC, nomCinéma
FROM Cinéma
WHERE EXISTS  (SELECT NumSalle
               FROM Salle
               WHERE numCSal=numC
               AND capacité >200) ;
```

Résultat attendu : les cinémas N° 1,…, 7, 9, 10, 14, 15 (remarque : ville ='Alger' est fausse, car dans certains tuples de la table Cinéma il ya un code postale associé au nom de la ville…. Il existe aussi le cas ou l'on introduit dans le champ ville la phrase : Ville d'Alger).

- Donner les cinémas qui se trouvent à Alger « **ET** » qui ont une salle de capacité supérieure à 200 places.

```
SELECT numC, nomCinéma
FROM Cinéma
```

```
WHERE ville LIKE '%Alger%'
INTERSECT
SELECT Cinéma.numC, Cinéma.nomCinéma
FROM Cinéma, Salle
WHERE Cinéma.numC= Salle.numCSal
AND  capacité >200 ;
```

La commande SQL INTERSECT permet **d'obtenir l'intersection des résultats de 2 requêtes**. Cette commande permet donc de récupérer les enregistrements communs à 2 requêtes. Pour l'utiliser convenablement il faut que les 2 requêtes retournent le même nombre de colonnes, avec les mêmes types et dans le même ordre. Cependant, cette commande est compatible seulement avec les SGBDR PostgreSQL, SQL Server, Oracle et SQLite. **Elle n'est pas disponible sous MySQL.**

On peut également écrire :

```
SELECT numC, nomCinéma
FROM Cinéma
WHERE ville LIKE '%Alger%'
AND numC IN  (SELECT numCSal
                FROM Salle
                WHERE capacité >200) ;
```

Résultat attendu : les cinémas N° 1, 3, 4, 5, 6, 7.

- Donner les cinémas sans salles.

Cinéma (**numC**, nomCinéma, numéro, rue, ville)
Salle (**NumSalle**, #numCSal, capacité, climatisée)
Film (**idFilm**, titre, année, genre, résumé)
Séance **(#idFilmS, #numCS, #NumSalleS**, tarif)

```
SELECT numC, nomCinéma
FROM Cinéma
WHERE numC NOT IN (SELECT numCSal
                     FROM Salle);
```

On peut également faire :

```
SELECT numC, nomCinéma
FROM Cinéma
WHERE NOT EXISTS (SELECT Salle.NumSalle
                    FROM Salle
                    WHERE Salle.numCSal = Cinéma.numC);
```

Résultat attendu : les cinémas N° 8,11, 12, 13, 16, 17, 18, 19, 20, 21.

- Donner les cinémas qui se trouvent à Constantine et possèdent des salles climatisée.

```
SELECT numC, nomCinéma
FROM Cinéma
WHERE ville LIKE '%Constantine%'
 AND numC IN (SELECT numCSal
                    FROM Salle
                    WHERE climatisée='TRUE');
```

On peut également faire :

```
SELECT numC, nomCinéma
FROM Cinéma
WHERE ville LIKE '%Constantine%'
AND EXISTS (SELECT Salle.NumSalle
                FROM Salle
                WHERE Salle.numCSal = Cinéma.numC AND climatisée='TRUE');
```

Résultat attendu : les cinémas N°14, 15.

- Donner les films qui se passent au cinéma d'Oran et dans des salles climatisées.

Cinéma (**numC**, nomCinéma, numéro, rue, ville)
Salle (**NumSalle**, #numCSal, capacité, climatisée)
Film (**idFilm**, titre, année, genre, résumé)
Séance **(#idFilmS, #numCS, #NumSalleS**, tarif)

```
SELECT idFilm, titre
FROM Film
WHERE idFilm IN (SELECT idFilmS
                        FROM Séance
                        WHERE NumSalleS IN (SELECT NumSalle
                                                FROM Cinéma, Salle
                                                WHERE Salle.numCSal=Cinéma.numC
                                                AND Salle.climatisée='TRUE'
                                                AND Cinéma.ville LIKE '%Oran%')) ;
```

On peut également faire :

```
SELECT idFilm, titre
FROM Film
WHERE idFilm IN (SELECT idFilmS
```

```
                    FROM Séance
                    WHERE NumSalleS IN (SELECT NumSalle
                                        FROM Salle
                                        WHERE climatisée='TRUE'
                                        AND numCSal IN (SELECT numC
                                                        FROM Cinéma
                                                        WHERE ville LIKE '%Oran%')));
```

Résultat attendu : les films N° 1, 6, 7, 8, 11

- Donner les films qui ont un tarif au moins supérieur à 350 DA

```
SELECT idFilm, titre
FROM Film
WHERE idFilm IN (SELECT idFilmS
                 FROM Séance
                 WHERE tarif >350) ;
```

On peut également faire :

```
SELECT Film.idFilm, Film.titre
FROM Film, Séance
WHERE Film.idFilm = Séance.idFilmS AND   Séance.tarif >350;
```

Résultat attendu : les films N° 7, 8, 9, 10, 11

- Donner les numéros des salles qui ne contiennent aucun film

```
SELECT NumSalle
FROM Salle
WHERE NumSalle NOT IN (SELECT NumSalleS
                       FROM Séance) ;
```

On peut également faire :

```
SELECT NumSalle
FROM Salle
WHERE NOT EXISTS (SELECT idFilmS, numCS, NumSalleS
                  FROM Séance
                  WHERE Séance.NumSalleS = Salle.NumSalle) ;
```

Résultat attendu : les salles N° 2,…,15, 18, 19, 22, 25, 26, 29, 30, 35, 45, 46, 49.

- Donner tous les films et leurs salles de projections

```
SELECT Film.idFilm, Film.titre, Séance.NumSalleS
FROM Film, Séance
WHERE Film.idFilm = Séance.idFilmS ;
```

- Donner tous les films et toutes les salles (c'est le produit cartésien).

```
SELECT Film.idFilm, Film.titre, Salle.NumSalle
FROM Film, Salle;
```