Scripts BD Q2

Labo 4 :

1. SELECT COUNT(\*)

FROM Candidat ;

1. SELECT COUNT(\*)

FROM Candidat

WHERE ca\_localite = 'Namur';

1. SELECT AVG(SYSDATE - datenaissance)/365

FROM Candidat

1. SELECT EXTRACT(YEAR FROM datenaissance) AS année, COUNT(\*) AS nbr\_candidat

FROM Candidat

GROUP BY EXTRACT(YEAR FROM datenaissance)

ORDER BY EXTRACT(YEAR FROM datenaissance) DESC;

1. SELECT j.titre, COUNT(p.datepostule) AS nombre\_candidat

FROM job j

LEFT OUTER JOIN postuler p ON j.id\_job = p.id\_job

GROUP BY j.titre

ORDER BY nombre\_candidat DESC;

1. SELECT pe.categorie, COUNT(c.dateobtention) AS nombre

FROM permisconduire pe

JOIN conduire c ON c.id\_permis = pe.id\_permis

GROUP BY pe.categorie;

1. SELECT MIN(ca\_codepostal), MAX(ca\_codepostal)

FROM candidat;

1. SELECT c.ca\_nom AS name, c.prenom AS fristname, e.ent\_nom as entreprise

FROM candidat c

JOIN postuler p ON c.id\_candidat = p.id\_candidat

JOIN job j ON p.id\_job = j.id\_job

JOIN entreprise e ON j.id\_entreprise = e.id\_entreprise;

1. SELECT cat.cat\_libelle, COUNT(p.datepostule)

FROM job j

JOIN categorie cat ON cat.id\_categorie = j.id\_categorie

JOIN postuler p ON p.id\_job = j.id\_job

GROUP BY cat.cat\_libelle;

1. SELECT c. id\_candidat, c.ca\_nom AS name

FROM candidat c

JOIN posseder p ON p.id\_candidat = c.id\_candidat

WHERE p.datediplome = (SELECT MAX(datediplome)

FROM posseder)

GROUP BY c.ca\_nom;

Ex perdu:

SELECT c. id\_candidat, c.ca\_nom, COUNT(p.id\_langue)

FROM parler p

JOIN candidat c ON c.id\_candidat = p.id\_candidat

GROUP BY c.ca\_nom

HAVING COUNT(p.id\_langue) >= 3;

1. SELECT c. id\_candidat, c.ca\_nom, COUNT(j.id\_horaire)

FROM candidat c

JOIN postuler p ON p.id\_candidat = c.id\_candidat

JOIN job j ON p.id\_job = j.id\_job

group by c.ca\_nom;

1. SELECT c. id\_candidat, c.ca\_nom, COUNT(j.id\_horaire)

FROM candidat c

JOIN postuler p ON p.id\_candidat = c.id\_candidat

JOIN job j ON p.id\_job = j.id\_job

group by c.ca\_nom

HAVING COUNT(j.id\_horaire) > 2;

1. SELECT c. id\_candidat, c.ca\_nom, COUNT(h.id\_horaire)

FROM candidat c

JOIN postuler p ON p.id\_candidat = c.id\_candidat

JOIN job j ON p.id\_job = j.id\_job

JOIN horaire h ON j.id\_horaire = h.id\_horaire

WHERE h.ho\_libelle = 'Mi-Temps'

GROUP BY c.ca\_nom

HAVING COUNT(h.id\_horaire) <= 2;

1. *SELECT C*.ID\_CANDIDAT, *C*.CA\_NOM, Q.QU\_LIBELLE  
   *FROM* CANDIDAT *C  
    JOIN* POSSEDER P *on C*.ID\_CANDIDAT = P.ID\_CANDIDAT  
    *JOIN* QUALIFICATION Q *on* P.ID\_QUALIFICATION = Q.ID\_QUALIFICATION  
   *WHERE* Q.ID\_QUALIFICATION *NOT IN* (*SELECT* QUAL.ID\_QUALIFICATION  
    *FROM* QUALIFICATION QUAL  
    *JOIN* NECESSITER N *on* QUAL.ID\_QUALIFICATION = N.ID\_QUALIFICATION  
    *JOIN JOB* J *on* N.ID\_JOB = J.ID\_JOB)
2. SELECT c.id\_candidat, c.ca\_nom AS name, p.datediplome

FROM candidat c

JOIN posseder p ON p.id\_candidat = c.id\_candidat

WHERE p.datediplome > (SELECT MAX(p.datediplome)

FROM candidat c

JOIN posseder p ON c.id\_candidat = p.id\_candidat

WHERE c.ca\_codepostal = 5000

) ;



*SELECT* CA\_NOM  
*FROM* CANDIDAT  
*WHERE* DATENAISSANCE = (*SELECT MAX*(DATENAISSANCE) *FROM* CANDIDAT)

1. SELECT c.ca\_nom, c.prenom, n.nat\_libelle

FROM candidat c

JOIN nationalite n ON c.id\_nationalite = n.id\_nationalite

WHERE n.nat\_libelle LIKE '%e%';

1. SELECT id\_candidat

FROM candidat

MINUS

SELECT id\_candidat

FROM conduire:

1. SELECT et.et\_nom AS etablissement, COUNT(p.id\_candidat) AS nombre

FROM posseder p

JOIN etablissement et ON p.id\_etablissement = et.id\_etablissement

GROUP BY et.et\_nom

ORDER BY COUNT(p.id\_candidat) DESC;

1. SELECT et.et\_nom AS etablissement, COUNT(p.id\_candidat) AS nombre

FROM posseder p

JOIN etablissement et ON p.id\_etablissement = et.id\_etablissement

GROUP BY et.et\_nom

HAVING COUNT(p.id\_candidat) = (SELECT MAX(COUNT(p.id\_candidat))

FROM posseder p

JOIN etablissement et ON p.id\_etablissement = et.id\_etablissement

GROUP BY et.et\_nom

);

2. SELECT ent.ent\_nom, COUNT(d.id\_avantage) AS nombre

FROM job j

JOIN entreprise ent ON j.id\_entreprise = ent.id\_entreprise

JOIN donner d ON j.id\_job = d.id\_job

GROUP BY ent.ent\_nom;

1. SELECT c.ca\_nom, c.prenom, p.datediplome

FROM candidat c

JOIN posseder p ON c.id\_candidat = p.id\_candidat

WHERE p.datediplome = (SELECT MAX(datediplome)

FROM posseder);

SELECT j.titre, tr.tr\_libelle AS tranche

FROM job j

JOIN tranchesalariale tr ON j.id\_tranchesalariale = tr.id\_tranchesalariale

UNION

SELECT titre, '2500-3000' AS tranche

FROM job

WHERE id\_tranchesalariale IS NULL

1. A)

SELECT j.titre, COUNT(p.id\_candidat)

FROM job j

JOIN postuler p ON j.id\_job = p.id\_job

GROUP BY j.titre

HAVING COUNT(p.id\_candidat) = 1;

B)

SELECT c.id\_candidat, c.ca\_nom as nom, c.prenom, COUNT(p.id\_job) AS nombre

FROM postuler p

JOIN candidat c ON c.id\_candidat = p.id\_candidat

GROUP BY c.id\_candidat, c.ca\_nom, c.prenom

HAVING COUNT(p.id\_job) = 1

ORDER BY c.id\_candidat;

C)

*SELECT* J.TITRE, *COUNT*(P.ID\_CANDIDAT) *AS* NOM  
*FROM JOB* J  
*JOIN* POSTULER P *on* J.ID\_JOB = P.ID\_JOB  
*JOIN* CANDIDAT *C on* P.ID\_CANDIDAT = *C*.ID\_CANDIDAT  
*WHERE C*.CA\_NOM *IN* (  
 *SELECT* C2.CA\_NOM  
 *FROM* CANDIDAT C2  
 *JOIN* POSTULER P2 *on* C2.ID\_CANDIDAT = P2.ID\_CANDIDAT  
 *GROUP BY* C2.CA\_NOM  
 *HAVING COUNT*(P2.ID\_JOB) = 1  
 )  
*GROUP BY* J.TITRE  
*HAVING COUNT*(P.ID\_CANDIDAT) = 1;

Labo 5:

1. UPDATE candidat

SET email = NULL

WHERE email NOT LIKE '\_@\_.\_';

ALTER TABLE candidat

ADD CONSTRAINT verification\_email

CHECK (email LIKE '\_@\_.\_');

1. ALTER TABLE parler

ADD CONSTRAINT niveau\_requis

CHECK (niveau BETWEEN 1 AND 5);

1. CREATE VIEW tranche\_salaire (titre, tr\_libelle)

AS (SELECT j.titre, tr.tr\_libelle

FROM job j

JOIN tranchesalariale tr ON j.id\_tranchesalariale = tr.id\_tranchesalariale

)

UNION

SELECT j.titre, '2500-3000'

FROM job j

WHERE j.id\_tranchesalariale IS NULL;

1. CREATE VIEW type\_job (titre, ent\_nom, contrat)

AS (SELECT j.titre, ent.ent\_nom, ty.ty\_code

FROM job j

JOIN entreprise ent ON j.id\_entreprise = ent.id\_entreprise

JOIN typecontrat ty ON j.id\_typecontrat = ty.id\_typecontrat

WHERE ty.ty\_code = 'CDI'

);

1. CREATE VIEW jobtempsplein

AS

SELECT \*

FROM job

WHERE id\_horaire = 1

WITH CHECK OPTION;

INSERT INTO jobtempsplein

VALUES ('26','responsable Monitoring','Gestion des serveurs grâce à Nagios et ITM (from IBM).',to\_date('31/01/2020','DD/MM/YYYY'),'16','1','7','5','2');

1. CREATE VIEW jobFlorennes (candidats, titre)

AS

SELECT COUNT(p.id\_candidat), j.titre

FROM job j

JOIN entreprise ent ON j.id\_entreprise = ent.id\_entreprise

JOIN postuler p ON j.id\_job = p.id\_job

GROUP BY j.titre, ent.ent\_nom

HAVING ent.ent\_nom LIKE 'ATHENEE ROYAL FLORENNES'

;



*CREATE VIEW* NON\_POSTULANTS  
*AS*(  
*SELECT* ID\_JOB  
*FROM JOB* j  
*MINUS  
SELECT* ID\_JOB  
*FROM* POSTULER  
);



*CREATE VIEW* candidats\_non\_postulants  
*AS*(  
*SELECT c*.ID\_CANDIDAT  
*FROM* CANDIDAT *c  
MINUS  
SELECT* p.ID\_CANDIDAT  
*FROM* POSTULER p  
);

1. FAIT

*CREATE VIEW* DATEDERNIERJOB (CA\_NOM, PRENOM, DATE\_POSTULER)  
*AS* (  
 *SELECT C*.CA\_NOM, *C*.PRENOM, *MAX*(P.DATEPOSTULE)  
 *FROM* CANDIDAT *C  
 JOIN* POSTULER P *on C*.ID\_CANDIDAT = P.ID\_CANDIDAT  
 *GROUP BY C*.CA\_NOM, *C*.PRENOM  
 )



*CREATE VIEW* TRANCHE\_SALARIALE (ID\_TRANCHE, VALEUR\_TRANCHE)  
*AS* (  
*SELECT* ID\_TRANCHESALARIALE,  
 *CASE  
 WHEN* ID\_TRANCHESALARIALE = 8 *THEN* 7500  
 *ELSE* (*TO\_NUMBER*(*SUBSTR*(TR\_LIBELLE, 1, 4)) + *TO\_NUMBER*(*SUBSTR*(TR\_LIBELLE, 6))) /2  
 *END AS* AVERAGE  
*FROM* TRANCHESALARIALE  
 )



V1

*CREATE VIEW* TRANCHE\_SALARIALE\_CATEGORIE (CATEGORIE, MOYENNE)  
*AS* (  
*SELECT C*.CAT\_LIBELLE, *ROUND*((*SUM*(TS.VALEUR\_TRANCHE) / *COUNT*(J.TITRE)), 0)  
*FROM* CATEGORIE *C  
JOIN JOB* J *on C*.ID\_CATEGORIE = J.ID\_CATEGORIE  
*JOIN* TRANCHESALARIALE TR *ON* J.ID\_TRANCHESALARIALE = TR.ID\_TRANCHESALARIALE  
*JOIN* TRANCHE\_SALARIALE TS *on* TR.ID\_TRANCHESALARIALE = TS.ID\_TRANCHE  
*GROUP BY C*.CAT\_LIBELLE  
 );

V2

*CREATE VIEW* TRANCHE\_SALARIALE\_CATEGORIE (CATEGORIE, MOYENNE)  
*AS* (  
*SELECT C*.CAT\_LIBELLE, *ROUND*(*AVG*(*TO\_NUMBER*(*SUBSTR*(T.TR\_LIBELLE, 1, 4)) + *TO\_NUMBER*(*SUBSTR*(T.TR\_LIBELLE, 6))) / 2, 0)  
*FROM* CATEGORIE *C  
JOIN JOB* J *on C*.ID\_CATEGORIE = J.ID\_CATEGORIE  
*JOIN* TRANCHESALARIALE T *on* J.ID\_TRANCHESALARIALE = T.ID\_TRANCHESALARIALE  
*GROUP BY C*.CAT\_LIBELLE  
 );

Labo 6 :

1. Niveau langue entre 1 et 5

*CREATE TRIGGER* niveau\_langue *ON* Parler  
 *AFTER UPDATE*, *INSERT  
 AS  
 BEGIN  
 DECLARE* @niveau *INT  
 DECLARE* curseur\_langue *CURSOR  
 FOR  
 SELECT* i.Niveau  
 *FROM* inserted i  
 *JOIN* Candidat c *ON* i.id\_Candidat = c.id\_candidat  
 *JOIN* Parler p *on* c.id\_candidat = p.id\_Candidat  
  
 *OPEN* curseur\_langue  
 *FETCH* curseur\_langue *INTO* @niveau  
  
 *WHILE @@FETCH\_STATUS* = 0  
 *BEGIN  
 IF* @niveau *NOT BETWEEN* 1 *AND* 5  
 *BEGIN  
 RAISERROR* ('Niveau non conforme', 10, 1)  
 *ROLLBACK TRANSACTION  
 RETURN  
 end  
 ELSE  
 BEGIN  
 PRINT* N'Niveau modifié'  
 *end  
 FETCH* curseur\_langue *INTO* @niveau  
 *end  
 CLOSE* curseur\_langue  
 *DEALLOCATE* curseur\_langue  
 *end  
  
DROP TRIGGER* niveau\_langue  
  
*UPDATE* Parler  
*SET* Niveau = 2  
*WHERE* id\_Candidat = 23

1. Date du permis après 18 ans. 2 versions.

*2.1*

*CREATE TRIGGER* verif\_date *ON* Conduire  
*AFTER INSERT*, *UPDATE  
AS  
BEGIN  
 DECLARE* @date\_obtention *date  
 DECLARE* @date\_naissance *date  
 DECLARE* @candidat *varchar*(50)  
  
 *DECLARE* cursor\_date\_obtention *CURSOR  
 FOR  
 SELECT* i.DateObtention, c.DateNaissance, c.Prenom  
 *FROM* inserted i  
 *JOIN* Candidat c *ON* i.id\_Candidat = c.id\_candidat  
  
 *OPEN* cursor\_date\_obtention  
 *FETCH* cursor\_date\_obtention *INTO* @date\_obtention, @date\_naissance, @candidat  
  
 *WHILE @@FETCH\_STATUS* = 0  
 *BEGIN  
 IF* (*datediff*(*yy*,@date\_naissance, @date\_obtention) < 18)  
 *BEGIN  
 RAISERROR*('%s a passe son permis trop jeune',10,1, @candidat)  
 *ROLLBACK TRANSACTION  
 end  
 ELSE  
 BEGIN  
 PRINT* @candidat + ' a obtenu son permis dans les regles'  
   
 *end*

*FETCH* cursor\_date\_obtention *INTO* @date\_obtention, @date\_naissance, @candidat *end  
 CLOSE* cursor\_date\_obtention  
 *DEALLOCATE* cursor\_date\_obtention  
*end  
  
  
DROP TRIGGER* verif\_date;  
  
  
*UPDATE* Conduire  
*SET* DateObtention = '28/01/2010'  
*WHERE* id\_Candidat = 23;  
  
  
  
*INSERT INTO* Candidat (id\_candidat, Ca\_Nom, Prenom, Ca\_Rue, Ca\_Numero, Ca\_Localite, Ca\_CodePostal, Tel, Email, DateNaissance, id\_Nationalite, id\_EC)  
*VALUES*(23, 'Bodson', 'Fabrice', 'Rue de la croix', '1', 'Aywaille', 4920, 0, 'fa.bodson@student.helmo.be', '28/01/2000', 1, 1);  
  
*INSERT INTO* Conduire (id\_Candidat, id\_Permis, DateObtention)  
*VALUES* (23, 3, '28/01/2018');  
  
*2.2*

*CREATE TRIGGER* age\_requis  
 *ON* Conduire  
 *AFTER INSERT*, *UPDATE  
 AS  
BEGIN  
 DECLARE* @date\_obtention *DATE  
 DECLARE* @date\_naissance *DATE  
 DECLARE* curseur\_date *CURSOR LOCAL  
 FOR  
 SELECT* i.dateobtention, c.datenaissance  
 *FROM* inserted i  
 *JOIN* Candidat c *ON* i.id\_Candidat = c.id\_candidat  
  
 *OPEN* curseur\_date  
 *FETCH* curseur\_date *INTO* @date\_obtention, @date\_naissance  
  
 *WHILE @@FETCH\_STATUS* = 0  
 *BEGIN  
 IF datediff*(*yy*, @date\_naissance, @date\_obtention) < 18  
 *BEGIN  
 RAISERROR* (N'Le candidat était trop jeune…', 10, 1)  
 *ROLLBACK TRANSACTION  
 RETURN  
 end  
 ELSE  
 BEGIN  
 PRINT* 'Permis obtenu dans les temps'  
 *end  
 FETCH* curseur\_date *INTO* @date\_obtention, @date\_naissance  
 *end  
 CLOSE* curseur\_date  
 *DEALLOCATE* curseur\_date  
*end  
  
DROP TRIGGER* age\_requis;  
  
*INSERT INTO* Conduire  
*VALUES* (23, 3, '18/10/2018');  
  
*UPDATE* Conduire  
*SET* DateObtention = '18/10/2019'  
*WHERE* id\_Candidat = 23;

1. Date de publication doit être après la date de public. actuelle si modifiée

*CREATE TRIGGER* modif\_publication  
 *ON Job  
 AFTER UPDATE  
 AS  
BEGIN  
 DECLARE* @nouvelle\_date\_publication *DATE  
 DECLARE* @ancienne\_date\_publication *DATE  
 DECLARE* curseur\_date\_publi *CURSOR LOCAL  
 FOR  
 SELECT* i.publie, *d*.publie  
 *FROM* inserted i  
 *JOIN* deleted *d ON d*.id\_Job = i.id\_Job  
  
 *OPEN* curseur\_date\_publi  
 *FETCH* curseur\_date\_publi *INTO* @nouvelle\_date\_publication, @ancienne\_date\_publication  
  
 *WHILE @@FETCH\_STATUS* = 0  
 *BEGIN  
 IF* @nouvelle\_date\_publication < @ancienne\_date\_publication  
 *BEGIN  
 RAISERROR* (N'La nouvelle date doit être plus tard que la date de publication actuelle', 10, 1)  
 *ROLLBACK TRANSACTION  
 RETURN  
 end  
 ELSE  
 BEGIN  
 PRINT* N'Date correcte'  
 *RETURN  
 end  
 FETCH* curseur\_date\_publi *INTO* @nouvelle\_date\_publication, @ancienne\_date\_publication  
 *end  
 CLOSE* curseur\_date\_publi  
 *DEALLOCATE* curseur\_date\_publi  
*end  
  
DROP TRIGGER* modif\_publication;  
  
*UPDATE Job  
SET* publie = '20/03/2020'  
*WHERE* id\_Job = 25;

1. Qualification requise

*CREATE TRIGGER* verif\_qualif  
 *ON* Postuler  
 *AFTER INSERT  
 AS  
BEGIN  
 -- Identifiants des qualifs --  
 DECLARE* @qualif\_candidat *INT  
 DECLARE* @qualif\_requise *INT  
  
 DECLARE* curseur\_qualif *CURSOR LOCAL  
 FOR  
 SELECT* poss.id\_Qualification, *N*.id\_Qualification  
 *FROM* Postuler P  
 *JOIN* Candidat C *on* P.id\_Candidat = C.id\_candidat  
 *JOIN* inserted i *ON* c.id\_candidat = i.id\_Candidat  
 *JOIN* Posseder poss *on* i.id\_candidat = poss.id\_Candidat  
 *JOIN Job* J *on* i.id\_Job = J.id\_Job  
 *JOIN* Necessiter *N on* J.id\_Job = *N*.id\_Job  
  
 *OPEN* curseur\_qualif  
 *FETCH* curseur\_qualif *INTO* @qualif\_candidat, @qualif\_requise  
 *WHILE @@FETCH\_STATUS* = 0  
 *BEGIN  
 IF* @qualif\_requise <> @qualif\_candidat  
 *BEGIN  
 RAISERROR* (N'La qualification du candidat ne correspond pas aux attentes.', 10, 1)  
 *ROLLBACK TRANSACTION  
 RETURN  
 end  
 ELSE  
 BEGIN  
 PRINT* N'Candidature acceptée'  
 *RETURN  
 end  
 FETCH* curseur\_qualif *INTO* @qualif\_candidat, @qualif\_requise  
 *end  
 CLOSE* curseur\_qualif  
 *DEALLOCATE* curseur\_qualif  
*end  
  
DROP TRIGGER* verif\_qualif;  
  
*INSERT INTO* Postuler  
*VALUES* (22, 24, '24/05/2020');

1. Supprimer avantage