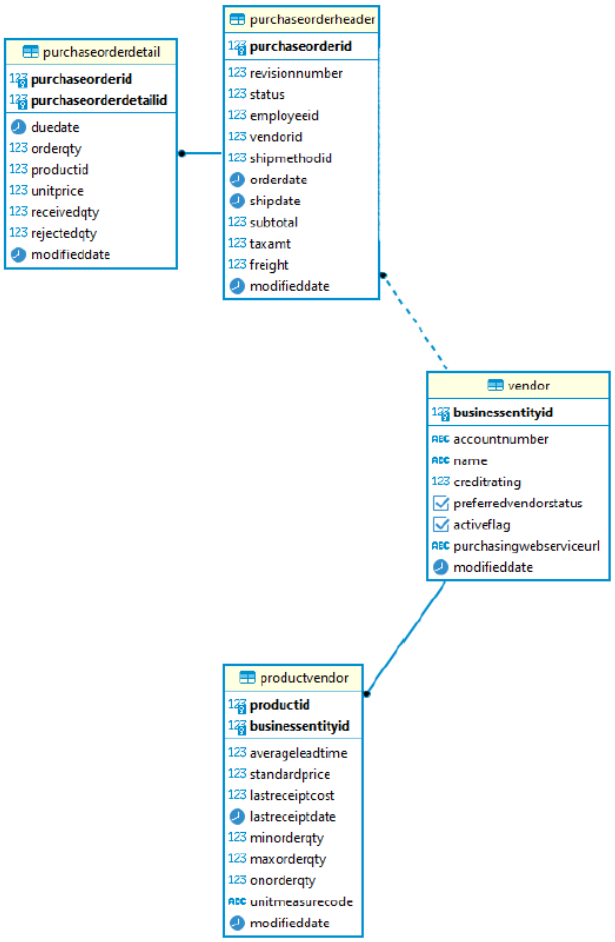
Projet final ADM – Order Management Model

Théophile NELSON, Eléonor KIOULOU, Khadija MOKHTARI – DIA4



Part1: Create tables and load data

1. Understand the relational data model.

2. Create the tables of the model by choosing the appropriate type of data for each column, (the data made available to you for each table can help you in this choice).

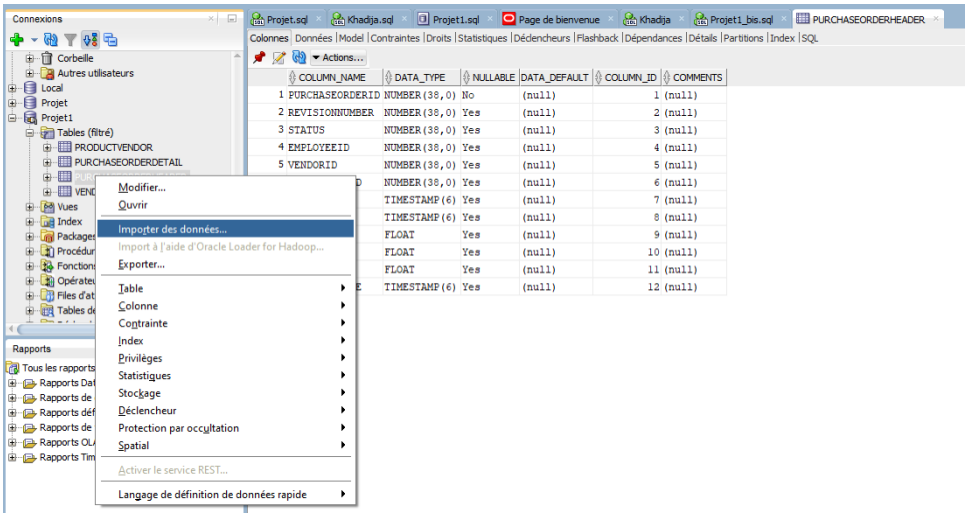
3. Implement the integrity constraints that you deem useful and/or relevant.

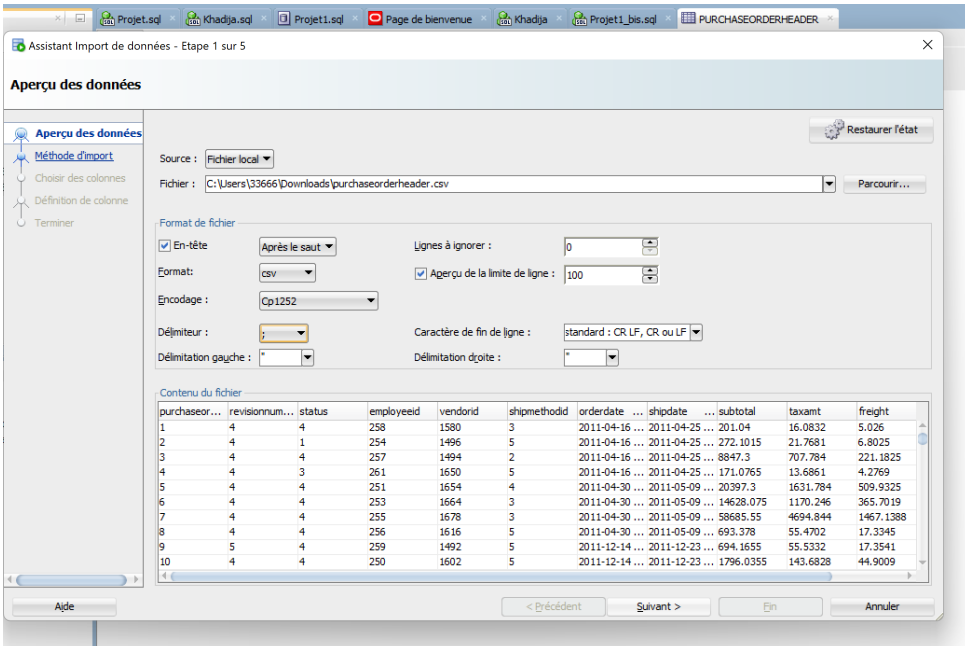
```
CREATE TABLE Vendor
 (
     businessentityid integer not null,
     accountnumber VARCHAR2(20),
     name VARCHAR2(100),
     creditrating integer,
     preferredvendorstatus VARCHAR2(50) CHECK( preferredvendorstatus IN ('false                ', 'true                 ') ),
     activeflag VARCHAR2(50) CHECK( activeflag IN ('false      ', 'true       ') ),
     purchasingwebserviceurl VARCHAR2(100),
     modifieddate TIMESTAMP,
     constraint vendor primary key(businessentityid)
 );



create table productvendor
 (
     productid integer not null,
     businessentityid integer not null,
     averageleadtime integer,
     standardprice float(100),
     lastreceiptcost float(100),
     lastreceiptdate timestamp,
     minorderqty integer,
     maxorderqty integer,
     onorderqty varchar2(30),
     unitmeasurecode varchar2(30),
     modifieddate timestamp,
     constraint productvendor_pk primary key (productid, businessentityid),
     constraint productvendor_fk2 foreign key (businessentityid) references vendor(businessentityid) ON DELETE CASCADE
 );
```

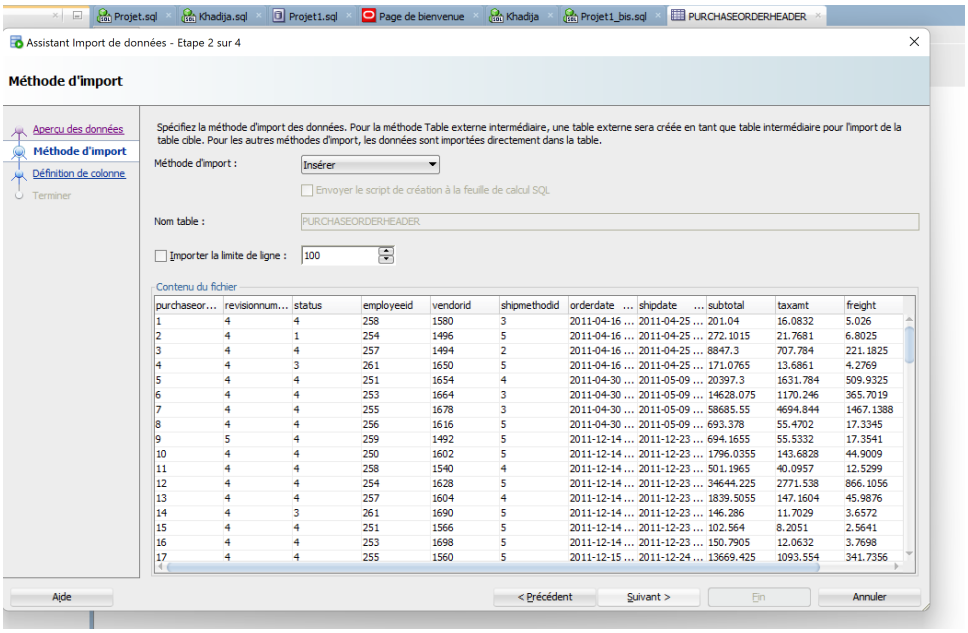
```
create table purchaseorderheader
 (
     purchaseorderid integer not null,
     revisionnumber integer,
     status integer,
     employeeid integer,
     vendorid integer,
     shipmethodid integer,
     orderdate timestamp,
     shipdate timestamp,
     subtotal float(30),
     taxamt float(30),
     freight float(30),
     modifieddate timestamp,
     constraint purchaseorderheader_pk primary key (purchaseorderid)
 );


create table purchaseorderdetail
 (
     purchaseorderid integer not null,
     purchaseorderdetailid integer not null,
     duedate timestamp,
     orderqty integer,
     productid integer,
     unitprice float(30),
     receivedqty integer,
     rejectedqty integer,
     modifieddate timestamp,
     constraint purchaseorderdetail_pk primary key (purchaseorderid,purchaseorderdetailid),
     constraint purchaseorderdetail_fk1 foreign key(purchaseorderid) references purchaseorderheader(purchaseorderid) ON DELETE CASCADE
 );
```

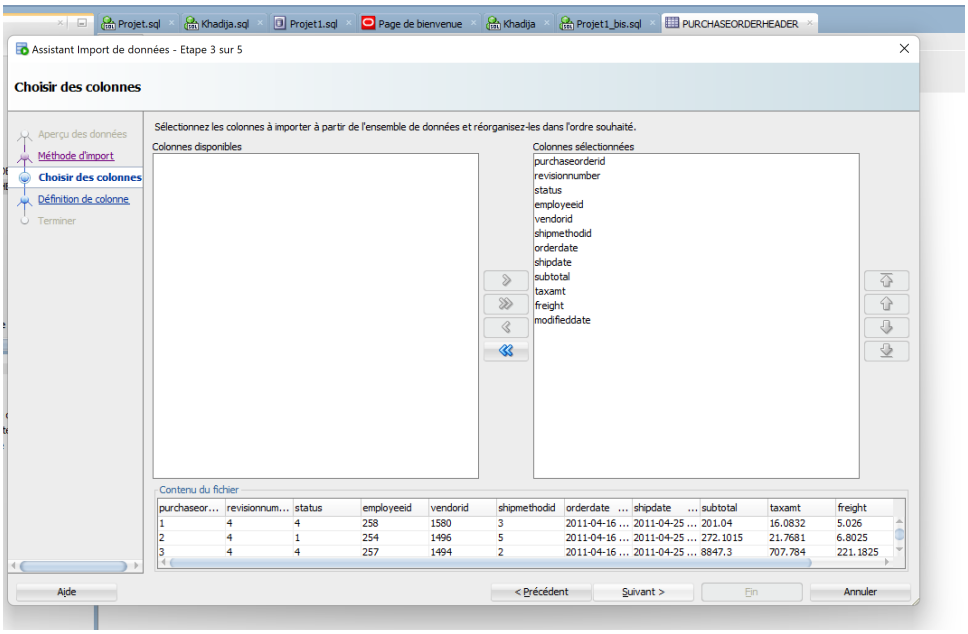
```
Table VENDOR créé(e).


Table PRODUCTVENDOR créé(e).


Table PURCHASEORDERHEADER créé(e).


Table PURCHASEORDERDETAIL créé(e).
```

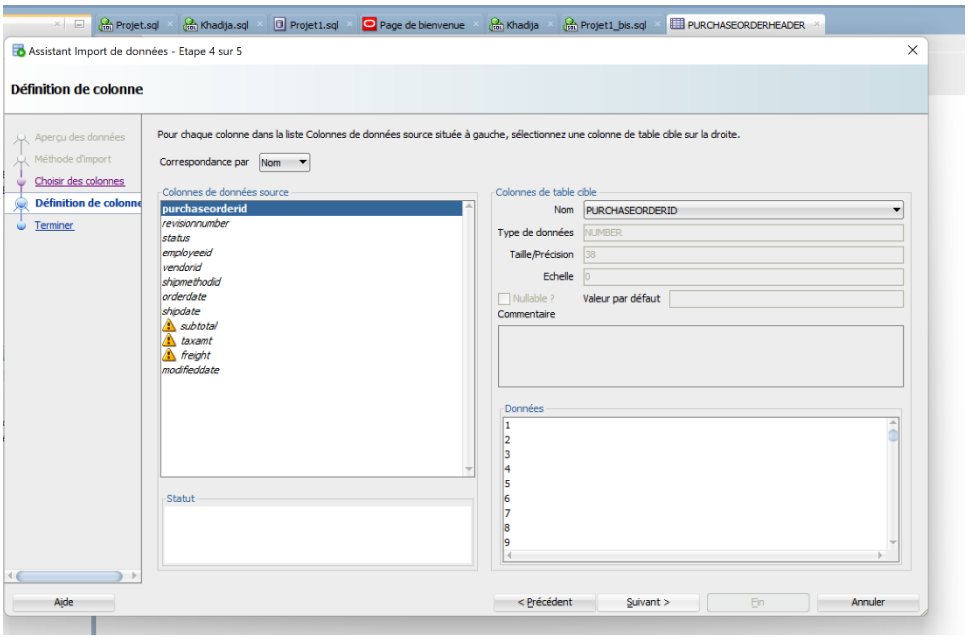
4. Develop a tool that allows you to load data into the model tables (by using a sql script, an Oracle utility (in this case SqlLoader), or import from SqlDeveloper tool … etc).

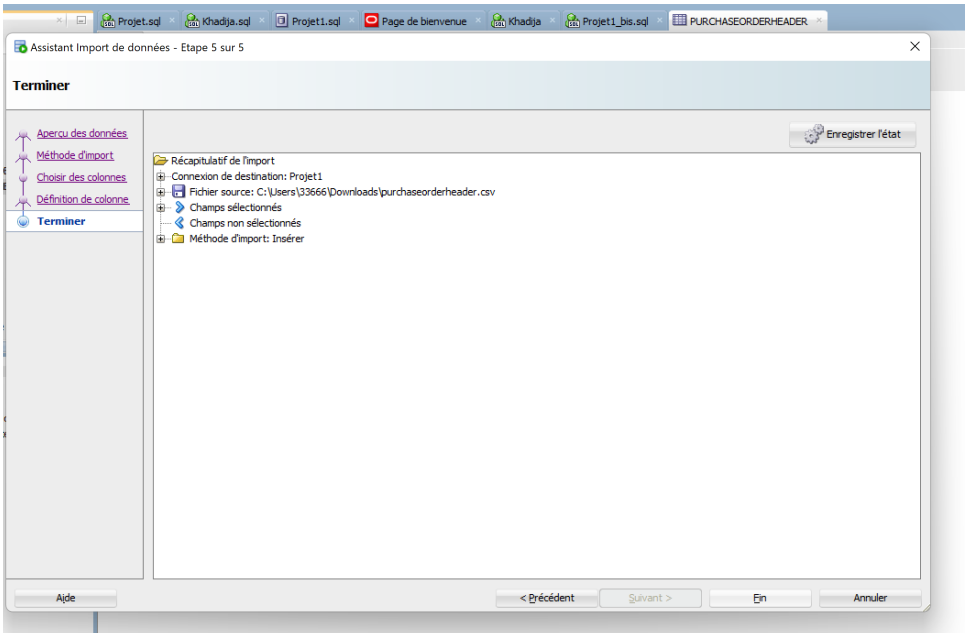


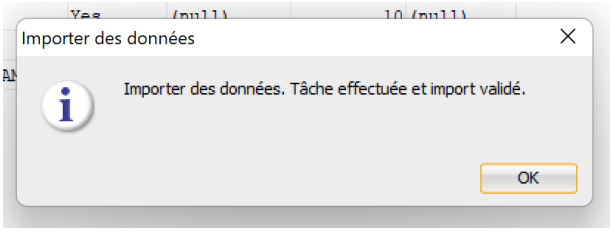



Projet.sql
Khadija.sql
Projet1.sql
Page de bienvenue
Khadija
Projet1_bis.sql
PURCHASEORDERHEADER
Assistant Import de données - Etape 2 sur 4
Méthode d'import
Aperçu des données
Méthode d'import
Définition de colonne
Terminer
Spécifiez la méthode d'import des données. Pour la méthode Table externe intermédiaire, une table externe sera créée en tant que table intermédiaire pour l'import de la table cible. Pour les autres méthodes d'import, les données sont importées directement dans la table.
Méthode d'import :
Insérer
Envoyer le script de création à la feuille de calcul SQL
Nom table :
PURCHASEORDERHEADER
Importer la limite de ligne :
100
Contenu du fichier
purchaseor…
revisionnum…
status
employeeid
vendorid
shipmethodid
orderdate …
shipdate …
subtotal
taxamt
freight
1 4 4 258 1580 3 2011-04-16 … 2011-04-25 … 201.04 16.0832 5.026
2 4 1 254 1496 5 2011-04-16 … 2011-04-25 … 272.1015 21.7681 6.8025
3 4 4 257 1494 2 2011-04-16 … 2011-04-25 … 8847.3 707.784 221.1825
4 4 3 261 1650 5 2011-04-16 … 2011-04-25 … 171.0765 13.6861 4.2769
5 4 4 251 1654 4 2011-04-30 … 2011-05-09 … 20397.3 1631.784 509.9325
6 4 4 253 1664 3 2011-04-30 … 2011-05-09 … 14628.075 1170.246 365.7019
7 4 4 255 1678 3 2011-04-30 … 2011-05-09 … 58685.55 4694.844 1467.1388
8 4 4 256 1616 5 2011-04-30 … 2011-05-09 … 693.378 55.4702 17.3345
9 5 4 259 1492 5 2011-12-14 … 2011-12-23 … 694.1655 55.5332 17.3541
10 4 4 250 1602 5 2011-12-14 … 2011-12-23 … 1796.0355 143.6828 44.9009
11 4 4 258 1540 4 2011-12-14 … 2011-12-23 … 501.1965 40.0957 12.5299
12 4 4 254 1628 5 2011-12-14 … 2011-12-23 … 34644.225 2771.538 866.1056
13 4 4 257 1604 4 2011-12-14 … 2011-12-23 … 1839.5055 147.1604 45.9876
14 4 3 261 1690 5 2011-12-14 … 2011-12-23 … 146.286 11.7029 3.6572
15 4 4 251 1566 5 2011-12-14 … 2011-12-23 … 102.564 8.2051 2.5641
16 4 4 253 1698 5 2011-12-14 … 2011-12-23 … 150.7905 12.0632 3.7698
17 4 4 255 1560 5 2011-12-15 … 2011-12-24 … 13669.425 1093.554 341.7356
Aide
< Précédent
Suivant >
Fin
Annuler


Projet.sql
Khadija.sql
Projet1.sql
Page de bienvenue
Khadija
Projet1_bis.sql
PURCHASEORDERHEADER
Assistant Import de données - Etape 3 sur 5
Choisir des colonnes
Aperçu des données
Méthode d'import
Choisir des colonnes
Définition de colonne
Terminer
Sélectionnez les colonnes à importer à partir de l'ensemble de données et réorganisez-les dans l'ordre souhaité.
Colonnes disponibles
Colonnes sélectionnées
purchaseorderid
revisionnumber
status
employeeid
vendorid
shipmethodid
orderdate
shipdate
subtotal
taxamt
freight
modifieddate
Contenu du fichier
purchaseor…
revisionnum…
status
employeeid
vendorid
shipmethodid
orderdate …
shipdate …
subtotal
taxamt
freight
1 4 4 258 1580 3 2011-04-16 … 2011-04-25 … 201.04 16.0832 5.026
2 4 1 254 1496 5 2011-04-16 … 2011-04-25 … 272.1015 21.7681 6.8025
3 4 4 257 1494 2 2011-04-16 … 2011-04-25 … 8847.3 707.784 221.1825
Aide
< Précédent
Suivant >
Fin
Annuler







We followed the same steps for each table of our database using the csv files given by our teacher and making sure we have started by the independent tables.

Part2: SQL queries

1. Write each of the following queries in SQL and try to optimize them. Improve query performance by using SQL query optimization tips and techniques (indexes and materialized views).

A. Display the Vendor names and the product numbers they sell for vendors with a credit rating of 5 and productid greater than 500.

```
--qA
select name , productid from vendor
inner join productvendor on vendor.businessentityid = productvendor.businessentityid
and vendor.creditrating = 5 and productvendor.productid > 500;
```

Résultat de req...

SQL | Toutes les lignes extraites : 4 en 0,339 secondes

	NAME	PRODUCTID
1	Victory Bikes	922
2	Victory Bikes	923
3	Victory Bikes	933
4	Victory Bikes	934

B. Display the purchase order number, OrderDate, purchase order detail id, order qty and product number for any purchase order with anOrder qty greater than 500.

```
90
91  -- Question B --
92  SELECT ph.purchaseorderid, ph.orderdate, pd.purchaseorderdetailid, pd.orderqty, pv.productid
93  FROM purchaseorderheader ph, purchaseorderdetail pd, productvendor pv
94  WHERE pd.purchaseorderid = ph.purchaseorderid
95  AND pd.productid = pv.productid
96  AND pd.orderqty > 500;
97
98
```

Sortie de script | Résultat de requête

SQL | 50 lignes extraites en 0,098 secondes

	PURCHASEORDERID	ORDERDATE	PURCHASEORDERDETAILID	ORDERQTY	PRODUCTID
1	269	17/10/12 00:00:00,000000000	626	550	510
2	269	17/10/12 00:00:00,000000000	626	550	510
3	269	17/10/12 00:00:00,000000000	627	550	511
4	269	17/10/12 00:00:00,000000000	627	550	511
5	269	17/10/12 00:00:00,000000000	628	550	512
6	269	17/10/12 00:00:00,000000000	628	550	512
7	270	17/10/12 00:00:00,000000000	629	550	916
8	270	17/10/12 00:00:00,000000000	629	550	916
9	271	17/10/12 00:00:00,000000000	630	550	525
10	272	17/10/12 00:00:00,000000000	631	550	526

C. Display the purchase order number, vendor number, purchase order detail id, product number and unit price. For purchase order numbers from 1400 to 1600.

```
--qC
select pod.purchaseorderid, v.accountnumber, pod.purchaseorderdetailid, pod.productid, pod.unitprice
from purchaseorderdetail pod
inner join productvendor pv on pv.productid = pod.productid
inner join vendor v on v.businessentityid = pv.businessentityid
and pod.purchaseorderid > = 1400 and pod.purchaseorderid < = 1600;
```

	PURCHASEORDERID	ACCOUNTNUMBER	PURCHASEORDERDETAILID	PRODUCTID	UNITPRICE
1	1403	LITWARE0001	3183	1	50,2635
2	1482	LITWARE0001	3348	1	50,2635
3	1561	LITWARE0001	3511	1	50,2635
4	1402	WOODFIT0001	3182	2	41,916
5	1481	WOODFIT0001	3347	2	41,916
6	1560	WOODFIT0001	3510	2	41,916
7	1406	AMERICAN0001	3187	4	57,0255
8	1485	AMERICAN0001	3353	4	57,0255
9	1564	AMERICAN0001	3515	4	57,0255
10	1409	VISIONC0001	3190	317	27,0585

...etc

832	1528	JACKSON0001	3435	939	48,2895
833	1535	JACKSON0001	3449	939	48,2895
834	1425	COMPETE0001	3224	940	62,9895
835	1504	COMPETE0001	3389	940	62,9895
836	1414	BICYCLE0001	3205	941	62,9895
837	1493	BICYCLE0001	3370	941	62,9895
838	1572	BICYCLE0001	3535	941	62,9895
839	1471	SUPERIOR0001	3326	948	82,8345
840	1550	SUPERIOR0001	3489	948	82,8345
841	1475	VARSITY0001	3333	952	15,7395
842	1554	VARSITY0001	3497	952	15,7395

D. Display how many orders are purchased from each vendor and the cost of the orders. Return the results sorted in descending order of highest cost.

```
101  -- Question D --
102  SELECT v.name, count(ph.purchaseorderid), sum(pd.unitprice) AS costorders
103  FROM vendor v, purchaseorderheader ph, purchaseorderdetail pd, productvendor pv
104  WHERE v.businessentityid = pv.businessentityid
105  AND pv.productid = pd.productid
106  AND pd.purchaseorderid = ph.purchaseorderid
107  GROUP BY v.name
108  ORDER BY costorders DESC;
109
```

Sortie de script | Résultat de requête

SQL | 50 lignes extraites en 0,169 secondes

	NAME	COUNT(PH.PURCHASEORDERID)	COSTORDERS
1	Mountain Works	678	28995,5085
2	Pro Sport Industries	560	24175,8405
3	Aurora Bike Center	560	24175,8405
4	Ready Rentals	531	22887,753
5	Australia Bike Retailer	531	22887,753
6	Leaf River Terrain	531	22887,753
7	Continental Pro Cycles	521	22562,211
8	Reliance Fitness, Inc.	521	22562,211
9	Speed Corporation	521	22562,211
10	Cruger Bike Company	523	21985,971

E. Display the average number of orders purchased across all vendors and the average cost across all vendors.

```
--qE
select Round(avg(orderqty), 2)orderqty,Round(avg(unitprice), 2)unitprice from purchaseorderdetail;
```

Résultat de requête | Résultat de requête 1

SQL | Toutes les lignes extraites : 1 en 0,083 secondes

	ORDERQTY	UNITPRICE
1	265,53	34,74

F. Display The top ten vendors with the highest percentage of rejected received items.

```
112  -- Question F --
113  SELECT v.name, count(pd.rejectedqty)/count(pd.receivedqty) AS rejectedreceiveditems
114  FROM vendor v, purchaseorderdetail pd, productvendor pv
115  WHERE v.businessentityid = pv.businessentityid
116  AND pv.productid = pd.productid
117  GROUP BY v.name
118  ORDER BY rejectedreceiveditems DESC
119  FETCH NEXT 10 ROWS ONLY;
120
```

Sortie de script × | Résultat de requête ×

SQL | Toutes les lignes extraites : 10 en 0,139 secondes

	NAME	REJECTEDRECEIVEDITEMS
1	Wide World Importers	1
2	International Bicycles	1
3	Electronic Bike Repair & Supplies	1
4	American Bikes	1
5	Gardner Touring Cycles	1
6	Chicago City Saddles	1
7	Federal Sport	1
8	First National Sport Co.	1
9	First Rate Bicycles	1
10	Expert Bike Co	1

G. Display The top ten vendors with the largest orders (in terms of quantity purchased).

```
-- qG
select v.name from vendor v
inner join productvendor pv on v.businessentityid = pv.businessentityid
group by pv.onorderqty, v.name
order by pv.onorderqty desc
FETCH FIRST 10 ROWS ONLY
```

Résultat de requête × | Résultat de requête 1 ×

SQL | Toutes les lignes extraites : 10 en 0,07 secondes

	NAME
1	Team Athletic Co.
2	Green Lake Bike Company
3	Jeff's Sporting Goods
4	Integrated Sport Products
5	Fitness Association
6	International Trek Center
7	Team Athletic Co.
8	Integrated Sport Products
9	Integrated Sport Products
10	Integrated Sport Products

H. Display the top ten products (in terms of quantity purchased).

```
123  -- Question H --
124  SELECT pv.productid, SUM(pd.orderqty) AS qtypurchased
125  FROM productvendor pv, purchaseorderdetail pd
126  WHERE pv.productid = pd.productid
127  GROUP BY pv.productid
128  ORDER BY qtypurchased DESC
129  FETCH NEXT 10 ROWS ONLY;
130
131
```

Sortie de script | Résultat de requête

SQL | Toutes les lignes extraites : 10 en 0,033 secondes

	PRODUCTID	QTYPURCHASED
1	319	214500
2	508	112200
3	524	112200
4	935	112200
5	523	112200
6	936	112200
7	507	112200
8	513	111100
9	510	101200
10	511	101200

I. Propose some complex sql queries using analytic functions

```
--qI
--Propose some complex sql queries using analytic functions

-- Display the vendor's name and its max order
select v.name, pv.maxorderqty from vendor v
inner join productvendor pv on v.businessentityid = pv.businessentityid
group by v.name, pv.maxorderqty
order by pv.maxorderqty desc
FETCH FIRST 1 ROWS ONLY
```

Résultat de requête | Résultat de requête 1

SQL | Toutes les lignes extraites : 1 en 0,079 secondes

	NAME	MAXORDERQTY
1	Jeff's Sporting Goods	15000

2. Create the two triggers below: J. Create a Transaction_History table with the same structure as PurchaseOrderDetail table. Implement using a trigger "After Update" On PurchaseOrderDetail table that :

- **Inserts a row in the Transaction_History table.**
- **Updates ModifiedDate in PurchaseOrderDetail.**

– **Updates the PurchaseOrderHeader.SubTotal column.**

```
DROP TRIGGER tg_after_update;
DROP TABLE transaction_history;
-- Question J --
CREATE TABLE transaction_history AS SELECT * FROM purchaseorderdetail;

CREATE TRIGGER tg_after_update
AFTER UPDATE ON purchaseorderdetail
FOR EACH ROW
BEGIN
    INSERT INTO transaction_history
    SELECT * FROM purchaseorderdetail
    WHERE purchaseorderdetailid = :NEW.purchaseorderdetailid;

    UPDATE purchaseorderdetail
    SET modifieddate = CURRENT_TIMESTAMP
    WHERE purchaseorderdetailid = :NEW.purchaseorderdetailid;

    UPDATE purchaseorderheader
    SET subtotal = (SELECT SUM(unitprice) FROM purchaseorderdetail WHERE purchaseorderid = :NEW.purchaseorderid)
    WHERE purchaseorderid = :NEW.purchaseorderid;
END;
/
```

Sortie de script | Résultat de requête

Tâche terminée en 0,31 secondes

```
Table TRANSACTION_HISTORY créé(e).


Elément Trigger TG_AFTER_UPDATE compilé
```

K. Implement using a trigger "Before Update" On PurchaseOrderHeader table that prohibits updates of the PurchaseOrderHeader.SubTotal column if the corresponding data in the PurchaseOrderDetail table is not consistent with the new value of the PurchaseOrderHeader.SubTotal column.

```
-- Question K --
CREATE TRIGGER tg_before_update
BEFORE UPDATE ON purchaseorderheader
FOR EACH ROW
DECLARE
    new_subtotal float(30);
    subtotal_detail float(30);
BEGIN
    SELECT SUM(unitprice * orderqty)
    INTO subtotal_detail
    FROM purchaseorderdetail
    WHERE purchaseorderid = :NEW.purchaseorderid;

    IF new_subtotal != subtotal_detail THEN
        RAISE_APPLICATION_ERROR(-2000, 'Cannot update purchase order subtotal: the corresponding data in the purchaseorderdetail
                                        table is not consistent with the new value of the purchaseorderheader.subtotal column.');
    END IF;
END;
/
```

Sortie de script | Résultat de requête

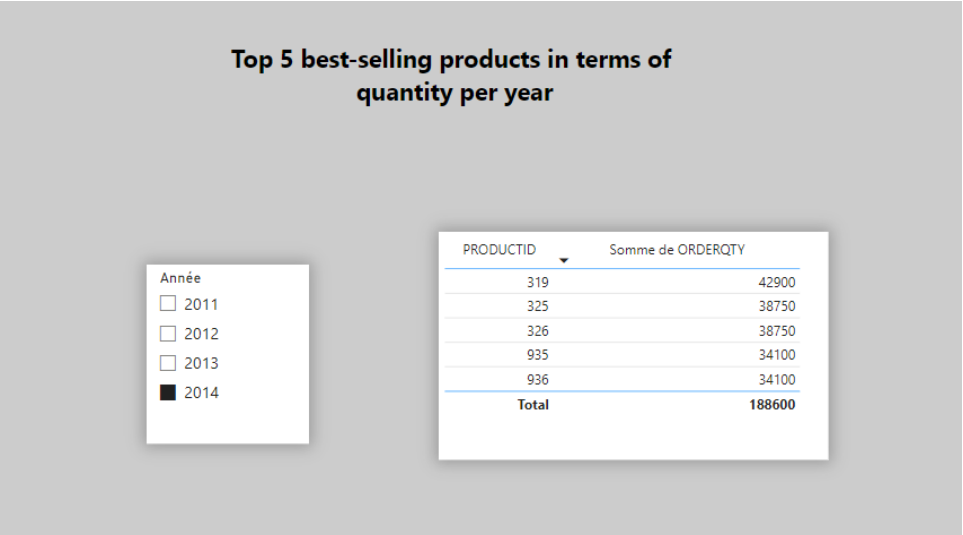
Tâche terminée en 0,076 secondes

```
Trigger TG_BEFORE_UPDATE supprimé(e).


Elément Trigger TG_BEFORE_UPDATE compilé
```

Part 3: Dashboards generating using Oracle and Bower BI

1. Generate the dashboards corresponding to the two questions below:

A - Generate a dashboard that represents the Top 5 best-selling products in terms of quantity per year.



B - Generate a dashboard that represents The Top 5 vendors who have the lowest percentage of rejected received items