SQL

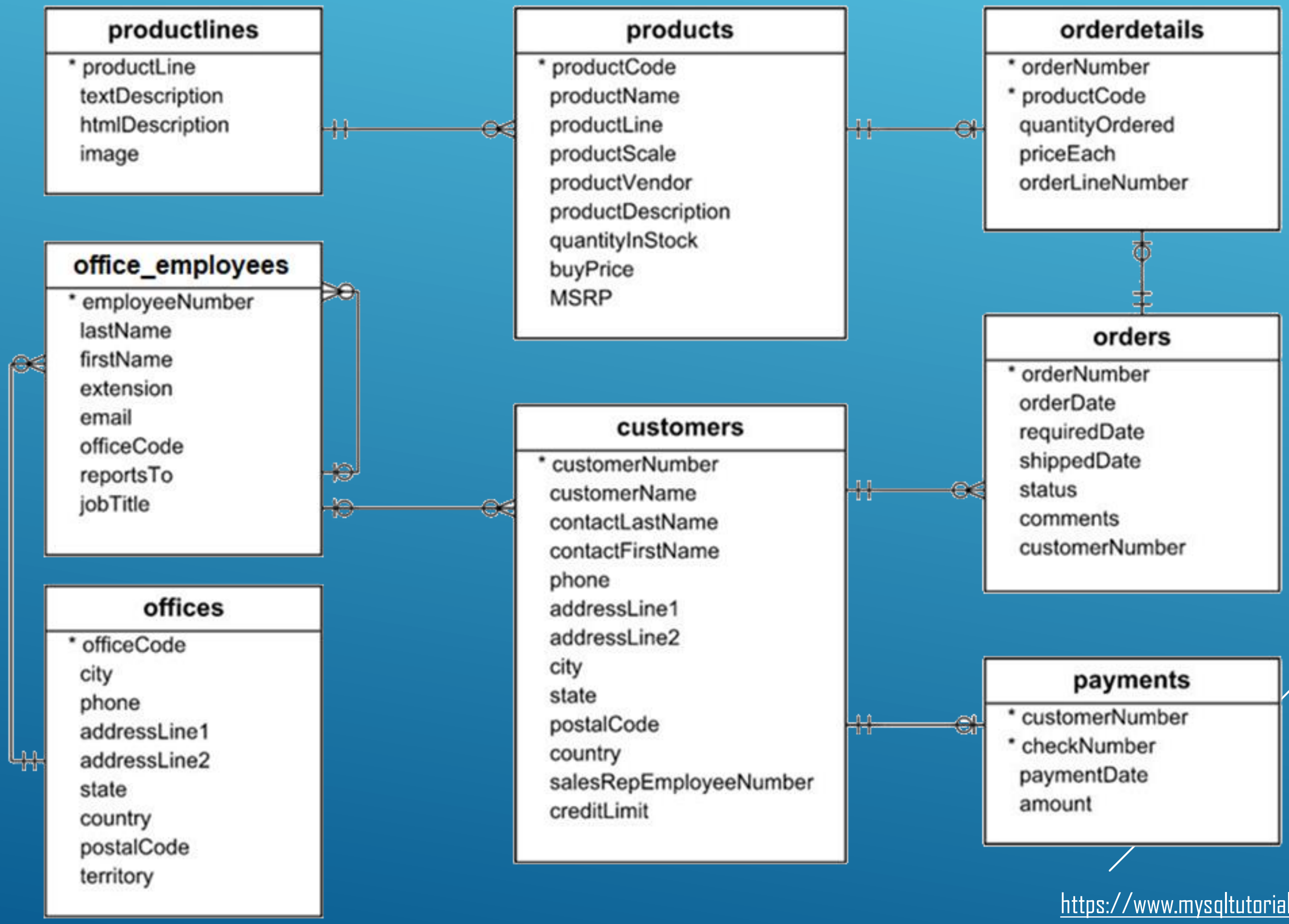
PART V (Grouping)

420-983-VA Introduction to Databases using Oracle Adrian Onet

SQL

Data Manipulation Language (DML)

Sample Models Schema. Describes an automotive models manufacturer and its sales.



https://www.mysqltutorial.org/mysql-sample-database.aspx

SQL

Data Manipulation Language (DML)

GROUPING

Grouping syntax.

```
SELECT fields1 FROM tables
     WHERE condition
  GROUP BY fields2
  HAVING condition;
```

SQL

Data Manipulation Language (DML)

GROUPING

Return the number of consultants that have an email. Schema details can be found here.

```
INSERT INTO consultants (employeeNumber, vendorEmail) VALUES (NULL, NULL);
```

employeeNumber	vendorEmail
1102	gbondur@vendors.com
1337	lbondur@vendors.com
1611	afixter@vendors.com
1625	ykato@vendors.com
NULL	abt@vendors.com
NULL	NULL

SQL

Data Manipulation Language (DML)

GROUPING

Return the number of consultants that have an email. Schema details can be found here.

```
SELECT count(*) AS total FROM consultants
     WHERE vendorEmail IS NOT NULL;
```

total
5

employeeNumber	vendorEmail
1102	gbondur@vendors.com
1337	lbondur@vendors.com
1611	afixter@vendors.com
1625	ykato@vendors.com
NULL	abt@vendors.com
NULL	NULL

What happens if we replace `count(*)` with:

a) `count(1)`

b) `count(employeeNumber)`

SQL

Data Manipulation Language (DML)

GROUPING

Return the number of consultants that have an email. Schema details can be found here.

```
SELECT count(*) AS total FROM consultants
    WHERE vendorEmail IS NOT NULL;
```

employeeNumber	vendorEmail
1102	gbondur@vendors.com
1337	lbondur@vendors.com
1611	afixter@vendors.com
1625	ykato@vendors.com
NULL	abt@vendors.com
NULL	NULL

total
5

What happens if we replace `count(*)` with:

a) `count(1)` ⟷ `count(*)`

b) `count(employeeNumber)` ⟶

total
4

SQL

Data Manipulation Language (DML)

GROUPING

Return the number of consultants that have an email. Schema details can be found here.

```
SELECT count(*) AS total FROM consultants
     WHERE vendorEmail IS NOT NULL;
```

employeeNumber	vendorEmail
1102	gbondur@vendors.com
1337	lbondur@vendors.com
1611	afixter@vendors.com
1625	ykato@vendors.com
NULL	abt@vendors.com
NULL	NULL

total
5

`count(expression)` counts number of rows for which expression evaluates to not NULL.

`count(*)` counts number of rows.

`count(DISTINCT expression)` counts number of rows with distinct value for expression that evaluates to not NULL.

SQL

Data Manipulation Language (DML)

GROUPING

Return the number of consultants that have an email. Schema details can be found here.

```
SELECT count(*) AS total FROM consultants
     WHERE vendorEmail IS NOT NULL;
```

employeeNumber	vendorEmail
1102	gbondur@vendors.com
1337	lbondur@vendors.com
1611	afixter@vendors.com
1625	ykato@vendors.com
NULL	abt@vendors.com
NULL	NULL

total
5

`count(expression)` counts number of rows for which expression evaluates to not NULL.

`count(*)` counts number of rows.

`count(DISTINCT expression)` counts number of rows with distinct value for expression that evaluates to not NULL.

`count(ALL expression)` ⟷ `count(expression)`

SQL

Data Manipulation Language (DML)

GROUPING

```
CREATE TABLE test (col INT NULL);
INSERT INTO test (col) VALUES (1);
INSERT INTO test (col) VALUES (1);
INSERT INTO test (col) VALUES (2);
INSERT INTO test (col) VALUES (3);
INSERT INTO test (col) VALUES (NULL);
INSERT INTO test (col) VALUES (NULL);
```

col
1
1
2
3
NULL
NULL

Check the result for the following queries:

```
SELECT count(1) AS total FROM test;

SELECT count(DISTINCT 1) AS total FROM test;

SELECT count(col) AS total FROM test;

SELECT count(DISTINCT col) AS total FROM test;

SELECT count(ALL col) AS total FROM test;
```

SQL

Data Manipulation Language (DML)

GROUPING

```
CREATE TABLE test (col INT NULL);
INSERT INTO test (col) VALUES (1);
INSERT INTO test (col) VALUES (1);
INSERT INTO test (col) VALUES (2);
INSERT INTO test (col) VALUES (3);
INSERT INTO test (col) VALUES (NULL);
INSERT INTO test (col) VALUES (NULL);
```

col
1
1
2
3
NULL
NULL

Check the result for the following queries:

```
SELECT count(1) AS total FROM test;
```

total
6

```
SELECT count(DISTINCT 1) AS total FROM test;
```

total
1

```
SELECT count(col) AS total FROM test;
```

total
4

```
SELECT count(DISTINCT col) AS total FROM test;
```

total
3

```
SELECT count(ALL col) AS total FROM test;
```

total
4

SQL

Data Manipulation Language (DML)

GROUPING

```
CREATE TABLE test (col INT NULL);
INSERT INTO test (col) VALUES (1);
INSERT INTO test (col) VALUES (1);
INSERT INTO test (col) VALUES (2);
INSERT INTO test (col) VALUES (3);
INSERT INTO test (col) VALUES (NULL);
INSERT INTO test (col) VALUES (NULL);
```

col
1
1
2
3
NULL
NULL

How do we return the distinct number of columns including NULLs , considering that NULLs should be counted only once? For the previous example should return 4.

SQL

Data Manipulation Language (DML)

GROUPING

```
CREATE TABLE test (col INT NULL);
INSERT INTO test (col) VALUES (1);
INSERT INTO test (col) VALUES (1);
INSERT INTO test (col) VALUES (2);
INSERT INTO test (col) VALUES (3);
INSERT INTO test (col) VALUES (NULL);
INSERT INTO test (col) VALUES (NULL);
```

col
1
1
2
3
NULL
NULL

How do we return the distinct number of columns including NULLs , considering that NULLs should be counted only once? For the previous example should return 4.

```
SELECT count(DISTINCT CASE WHEN col IS NULL THEN -1 ELSE col END) AS total FROM test;
```

```
SELECT count(DISTINCT IFNULL(col,-1)) AS total FROM test;
```

SQL

Data Manipulation Language (DML)

GROUPING

STUDENTS

student_id	name	gender
1	John	M
2	Adam	M
3	Sandra	F

BORROWED

student_id	book_id
1	id100
1	id200
3	id200
1	Id206
3	id200

BOOKS

book_id	author	title
id100	Ullman	DBMS
id200	Linz	Automata
id206	Baader	Term Rew.

Consider schema from here. Display all student id's together with the number of distinct books they borrowed.

SQL

Data Manipulation Language (DML)

GROUPING

STUDENTS

student_id	name	gender
1	John	M
2	Adam	M
3	Sandra	F

BORROWED

student_id	book_id
1	id100
1	id200
3	id200
1	Id206
3	id200

BOOKS

book_id	author	title
id100	Ullman	DBMS
id200	Linz	Automata
id206	Baader	Term Rew.

Consider schema from here. Display all student id's together with the number of distinct books they borrowed.

```
SELECT student_id, count(DISTINCT book_id) AS total
     FROM borrowed
     GROUP BY student_id;
```

student_id	total
1	3
3	1

Is this correct?

SQL

Data Manipulation Language (DML)

GROUPING

STUDENTS

student_id	name	gender
1	John	M
2	Adam	M
3	Sandra	F

BORROWED

student_id	book_id
1	id100
1	id200
3	id200
1	Id206
3	id200

BOOKS

book_id	author	title
id100	Ullman	DBMS
id200	Linz	Automata
id206	Baader	Term Rew.

Consider schema from here. Display all student id's together with the number of distinct books they borrowed.

```
SELECT student_id, count(DISTINCT book_id) AS total
     FROM borrowed
     GROUP BY student_id
UNION
(SELECT student_id, 0 FROM students
EXCEPT
SELECT student_id, 0 FROM borrowed);
```

ORACLE
MINUS

student_id	total
1	3
3	1
2	0

SQL

Data Manipulation Language (DML)

GROUPING

STUDENTS

student_id	name	gender
1	John	M
2	Adam	M
3	Sandra	F

BORROWED

student_id	book_id
1	id100
1	id200
3	id200
1	Id206
3	id200

BOOKS

book_id	author	title
id100	Ullman	DBMS
id200	Linz	Automata
id206	Baader	Term Rew.

Consider schema from here. Display all student id's together with the number of distinct books they borrowed.

```
SELECT student_id, count(DISTINCT book_id) AS total
     FROM borrowed
     GROUP BY student_id
UNION
SELECT S.student_id, 0
     FROM students S LEFT JOIN borrowed B ON S.student_id = B.student_id
     WHERE B.student_id IS NULL;
```

student_id	total
1	3
3	1
2	0

SQL

Data Manipulation Language (DML)

GROUPING

STUDENTS

student_id	name	gender
1	John	M
2	Adam	M
3	Sandra	F

BORROWED

student_id	book_id
1	id100
1	id200
3	id200
1	Id206
3	id200

BOOKS

book_id	author	title
id100	Ullman	DBMS
id200	Linz	Automata
id206	Baader	Term Rew.

Consider schema from here. Display all student id's that borrowed at least 2 books, not necessarily distinct.

SQL

Data Manipulation Language (DML)

GROUPING

STUDENTS

student_id	name	gender
1	John	M
2	Adam	M
3	Sandra	F

BORROWED

student_id	book_id
1	id100
1	id200
3	id200
1	Id206
3	id200

BOOKS

book_id	author	title
id100	Ullman	DBMS
id200	Linz	Automata
id206	Baader	Term Rew.

Consider schema from here. Display all student id's that borrowed at least 2 books, not necessarily distinct.

```
SELECT student_id, count(book_id) AS total
     FROM borrowed
     GROUP BY student_id
     HAVING count(book_id)>=2;
```

student_id	total
1	3
3	2

SQL

Data Manipulation Language (DML)

GROUPING

STUDENTS

student_id	name	gender
1	John	M
2	Adam	M
3	Sandra	F

BORROWED

student_id	book_id
1	id100
1	id200
3	id200
1	Id206
3	id200

BOOKS

book_id	author	title
id100	Ullman	DBMS
id200	Linz	Automata
id206	Baader	Term Rew.

Consider schema from here. Display all students name together with the number of distinct books they borrowed.

SQL

Data Manipulation Language (DML)

GROUPING

STUDENTS

student_id	name	gender
1	John	M
2	Adam	M
3	Sandra	F

BORROWED

student_id	book_id
1	id100
1	id200
3	id200
1	Id206
3	id200

BOOKS

book_id	author	title
id100	Ullman	DBMS
id200	Linz	Automata
id206	Baader	Term Rew.

Consider schema from here. Display all students name together with the number of distinct books they borrowed.

```
SELECT S.name, count(DISTINCT B.book_id) AS total
     FROM students S LEFT JOIN borrowed B
                    ON S.student_id = B.student_id
     GROUP BY S.student_id;
```

name	total
John	3
Adam	0
Sandra	1

SQL

Data Manipulation Language (DML)

GROUPING

STUDENTS

student_id	name	gender
1	John	M
2	Adam	M
3	Sandra	F

BORROWED

student_id	book_id
1	id100
1	id200
3	id200
1	Id206
3	id200

BOOKS

book_id	author	title
id100	Ullman	DBMS
id200	Linz	Automata
id206	Baader	Term Rew.

Consider schema from here. Return a table with 2 columns and 1 row. First column will display number of Female students and second column will display number of Male students.

$\longrightarrow$

female	male
1	2

SQL

Data Manipulation Language (DML)

GROUPING

STUDENTS

student_id	name	gender
1	John	M
2	Adam	M
3	Sandra	F

BORROWED

student_id	book_id
1	id100
1	id200
3	id200
1	Id206
3	id200

BOOKS

book_id	author	title
id100	Ullman	DBMS
id200	Linz	Automata
id206	Baader	Term Rew.

Consider schema from here. Return a table with 2 columns and 1 row. First column will display number of Female students and second column will display number of Male students.

```
SELECT
     SUM(CASE WHEN gender='F' THEN 1 ELSE 0 END) as female,
     SUM(CASE WHEN gender='M' THEN 1 ELSE 0 END) as male
FROM students;
```

female	male
1	2

SQL

Data Manipulation Language (DML)

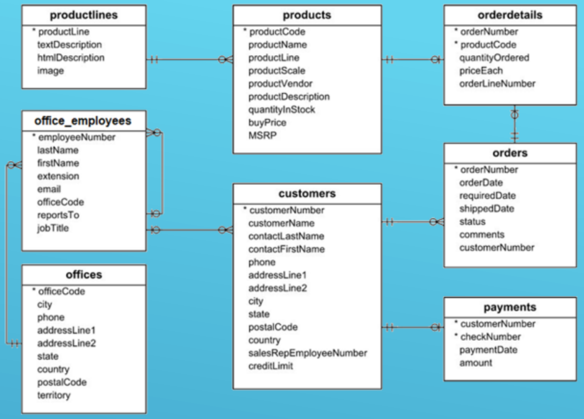
GROUPING (Aggregation Functions)

- **AVG** – average of the values in the group.
- **MIN** – minimum value from the group.
- **MAX** – maximum value from the group.
- **SUM** – sum of all values from the group.
- **STDEV** - statistical standard deviation of all values from the group.
- **COUNT** – counts the number of values in the group.

SQL

Data Manipulation Language (DML)

GROUPING (Problem)

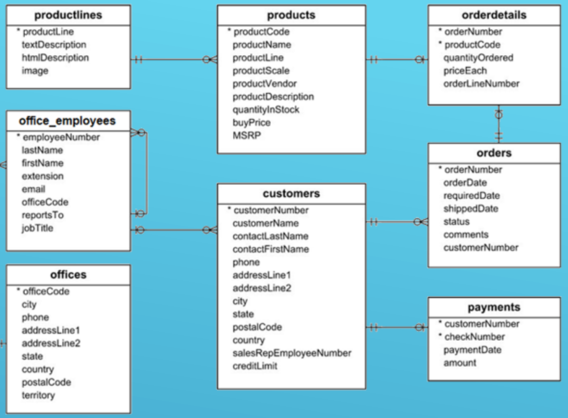


Return all product codes that were ordered with different prices (i.e. there are at least 2 orders with different priceEach value). Schema details can be found here.

SQL

Data Manipulation Language (DML)

GROUPING (Problem)



Return all product codes that were ordered with different prices (i.e. there are at least 2 orders with different priceEach value). Schema details can be found here.

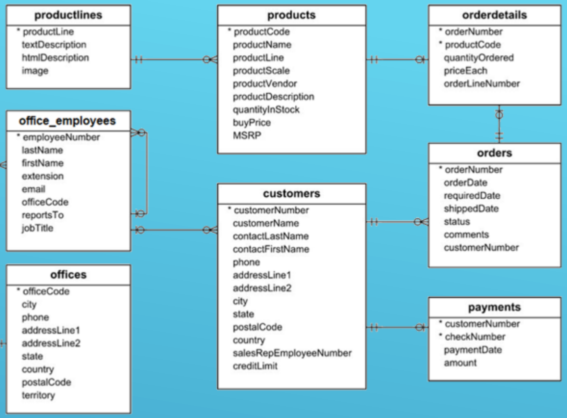
Solution 1. Not ideal.

```
SELECT DISTINCT O1.productCode
FROM orderDetails O1 INNER JOIN orderDetails O2
          ON O1.productCode=O2.productCode AND O1.priceEach<>O2.priceEach   ;
```

SQL

Data Manipulation Language (DML)

GROUPING (Problem)



Return all product codes that were ordered with different prices (i.e. there are at least 2 orders with different priceEach value). Schema details can be found here.

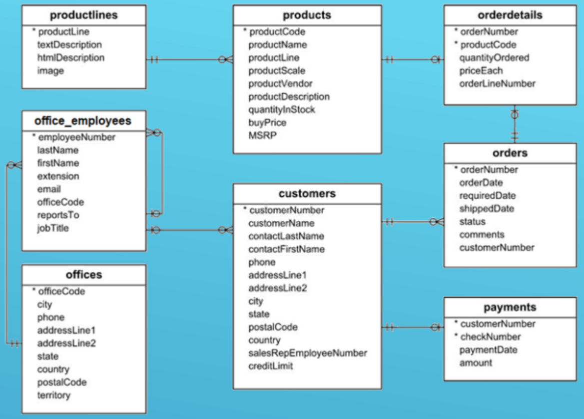
Solution 2. Not ideal.

```
SELECT productCode
     FROM orderDetails
     GROPUP BY productCode
     HAVING MIN(priceEach)<>MAX(priceEach);
```

SQL

Data Manipulation Language (DML)

GROUPING (Problem)



Return all product codes that were ordered with different prices (i.e. there are at least 2 orders with different priceEach value). Schema details can be found here.

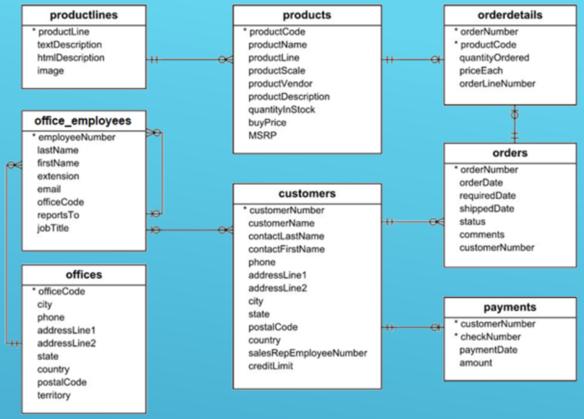
Solution 3.

```
SELECT productCode
     FROM orderDetails
     GROPUP BY productCode
     HAVING COUNT(DISTINCT priceEach)>=2;
```

SQL

Data Manipulation Language (DML)

ORDERING RESULTS

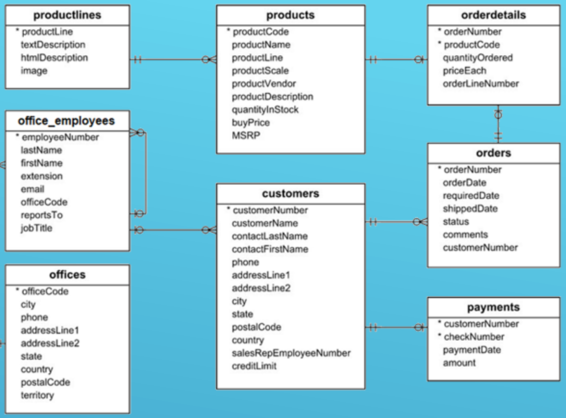


For each product and for each ordered price (orderDetails.priceEach) display product name, ordered price, total quantity ordered and total number of orders for those products with buyPrice greater than 50. Return data ordered by product name and for the same product ordered descending on priceEach. Schema details can be found here.

SQL

Data Manipulation Language (DML)

ORDERING RESULTS



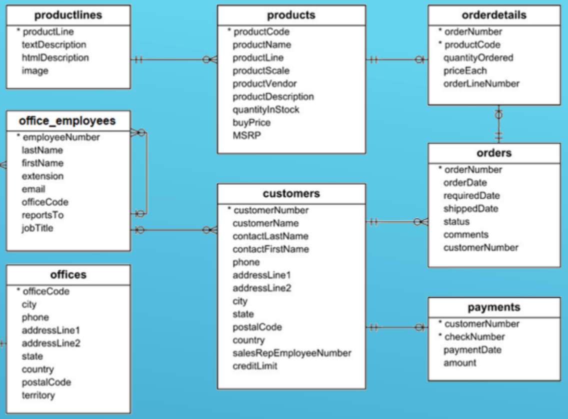
For each product and for each ordered price (orderDetails.priceEach) display product name, ordered price, total quantity ordered and total number of orders for those products with buyPrice greater than 50. Return data ordered by product name and for the same product ordered descending on priceEach. Schema details can be found here.

```
SELECT
     MIN(P.productName) AS productName,
     O.priceEach,
     SUM(O.quantityOrdered) AS quantity,
     COUNT(*) as numberOrders
FROM products P LEFT JOIN orderDetails O ON P.productCode=O.productCode
     WHERE P.buyPrice>50
     GROUP BY P.productCode, O.priceEach
     ORDER BY productName ASC, priceEach DESC;
```

SQL

Data Manipulation Language (DML)

ORDERING RESULTS



For each product and for each ordered price (orderDetails.priceEach) display product name, ordered price, total quantity ordered and total number of orders for those products with buyPrice greater than 50. Return data ordered by product name and for the same product ordered descending on priceEach. Schema details can be found here.

```
SELECT
     MIN(P.productName) AS productName,
     O.priceEach,
     SUM(O.quantityOrdered) AS quantity,
     COUNT(*) as numberOrders
FROM products P LEFT JOIN orderDetails O ON P.productCode=O.productCode
     WHERE P.buyPrice>50
     GROUP BY P.productCode, O.priceEach
     ORDER BY productName ASC, priceEach DESC;
```

```
     ORDER BY 1 ASC, 2 DESC
```

SQL

Data Manipulation Language (DML)

PIVOT

account	spending_date	amount
MasterCard	1/10/2020	100
Visa	1/12/2020	150
MasterCard	1/20/2020	100
MasterCard	5/14/2020	500
Visa	7/09/2020	299

For below schema, return a table with account as the first column. The rest of the columns will be named by the month of the spending date and corresponding value will be the total spent on that account for the given month.

Access

```
CREATE TABLE spendings (account VARCHAR(20), spending_date DATE, amount INT);
INSERT INTO spendings (account, spending_date, amount) VALUES ('MasterCard',#1/10/2020#, 100);
INSERT INTO spendings (account, spending_date, amount) VALUES ('Visa',#1/12/2020#, 150);
INSERT INTO spendings (account, spending_date, amount) VALUES ('MasterCard',#1/20/2020#, 100);
INSERT INTO spendings (account, spending_date, amount) VALUES ('MasterCard',#5/14/2020#, 500);
INSERT INTO spendings (account, spending_date, amount) VALUES ('Visa',#7/09/2020#, 299);
```

Result

account	1	5	7
MasterCard	200	500	
Visa	150		299

SQL

Data Manipulation Language (DML)

PIVOT

account	spending_date	amount
MasterCard	1/10/2020	100
Visa	1/12/2020	150
MasterCard	1/20/2020	100
MasterCard	5/14/2020	500
Visa	7/09/2020	299

For below schema, return a table with account as the first column. The rest of the columns will be named by the month of the spending date and corresponding value will be the total spent on that account for the given month.

Access

```
TRANSFORM SUM(amount)
 SELECT account FROM spendings GROUP BY account
PIVOT month(spending_date)
```

Result

account	1	5	7
MasterCard	200	500	
Visa	150		299

SQL

Data Manipulation Language (DML)

PIVOT

account	spending_date	amount
MasterCard	1/10/2020	100
Visa	1/12/2020	150
MasterCard	1/20/2020	100
MasterCard	5/14/2020	500
Visa	7/09/2020	299

For below schema, return a table with account as the first column. The rest of the columns will be named by the month of the spending date and corresponding value will be the total spent on that account for the given month.

```
CREATE TABLE spendings (account VARCHAR(20), m INT, amount INT);

INSERT INTO spendings (account, m, amount)
     VALUES ('MasterCard', 1, 100);

INSERT INTO spendings (account, m, amount)
     VALUES ('Visa', 1, 150);

INSERT INTO spendings (account, m, amount)
     VALUES ('MasterCard', 1, 100);

INSERT INTO spendings (account, m, amount)
     VALUES ('MasterCard', 5, 500);

INSERT INTO spendings (account, m, amount)
     VALUES ('Visa', 7, 299);
```

SQL

Data Manipulation Language (DML)

PIVOT

account	m	amount
MasterCard	1	100
Visa	1	150
MasterCard	1	100
MasterCard	5	500
Visa	7	299

For below schema, return a table with account as the first column. The rest of the columns will be named by the month of the spending date and corresponding value will be the total spent on that account for the given month.

```
SELECT * FROM spendings PIVOT ( SUM(amount) FOR m IN (1,2,3,4,5,6,7,8,9,10,11,12) )
```

Can be a subquery.

Groups by all fields except from the ones mentioned in the pivot clause.

Select done by grouping on account and month.

account	1	2	3	4	5	6	7	8	9	10	11	12
MasterCard	200	(null)	(null)	(null)	500	(null)	(null)	(null)	(null)	(null)	(null)	(null)
Visa	150	(null)	(null)	(null)	(null)	(null)	299	(null)	(null)	(null)	(null)	(null)

SQL

Data Manipulation Language (DML)

UNPIVOT

account	1	2	3	4	5	6	7	8	9	10	11	12
MasterCard	200	(null)	(null)	(null)	500	(null)	(null)	(null)	(null)	(null)	(null)	(null)
Visa	150	(null)	(null)	(null)	(null)	(null)	299	(null)	(null)	(null)	(null)	(null)

pivTable.

For below schema, return a table with account as the first column. The rest of the columns will be named by the month of the spending date and corresponding value will be the total spent on that account for the given month.

```
SELECT * FROM pivTable UNPIVOT
  ( amount FOR m IN ("1","2","3","4","5","6","7","8","9","10","11","12") )
```

account	m	amount
MasterCard	1	200
Visa	1	150
MasterCard	5	500
Visa	7	299