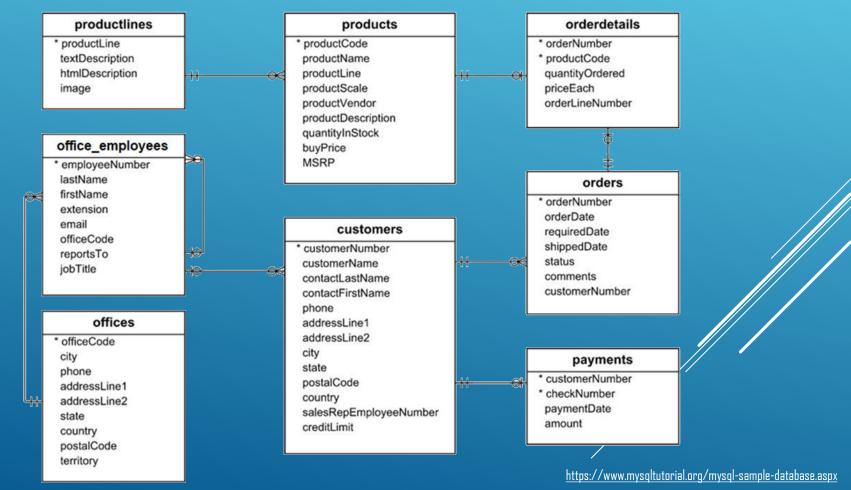
PART IV (Nested Queries)



### Data Manipulation Language (DML)

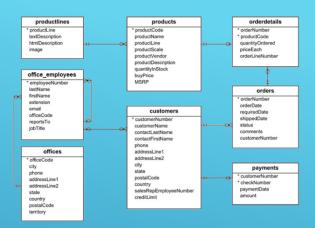
Sample Models Schema. Describes an automotive models manufacturer and its sales.



SQL

Data Manipulation Language (DML)

Nested Queries (IN)



Return all customers who made at least one payment greater than 3000\$. Schema details can be found <u>here</u>.

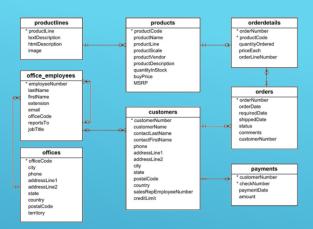


```
SELECT customerNumber, customerName FROM customers
WHERE customerNumber IN (SELECT customerNumber FROM payments WHERE amount>3000)
```

SQL

Data Manipulation Language (DML)

Nested Queries (IN)



Return all customers who made at least one payment greater than 3000\$. Schema details can be found <u>here</u>.



```
SELECT customerNumber, customerName FROM customers
WHERE customerNumber IN (SELECT customerNumber FROM payments WHERE amount>3000)
```

### Are these queries equivalent?

```
SELECT C.customerNumber, C.customerName

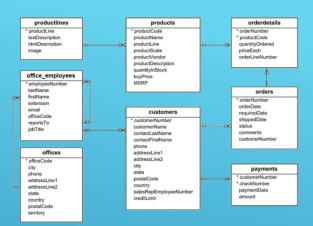
FROM customers C INNER JOIN payments P

ON C.customerNumber=P.customerNumber AND amount>3000
```

SQL

Data Manipulation Language (DML)

Nested Queries (IN)



Return all customers who made at least one payment greater than 3000\$. Schema details can be found here.



```
SELECT customerNumber, customerName FROM customers
WHERE customerNumber IN (SELECT customerNumber FROM payments WHERE amount>3000)
```

NO! In this case you may use DISTINCT but does not work in all cases.

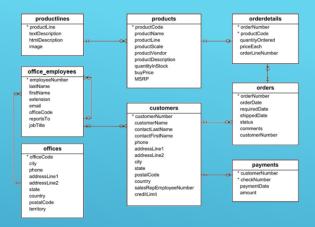
```
FROM customers C INNER JOIN payments P

ON C.customerNumber=P.customerNumber AND amount>3000
```

## SQL

Data Manipulation Language (DML)

Nested Queries (NOT IN)

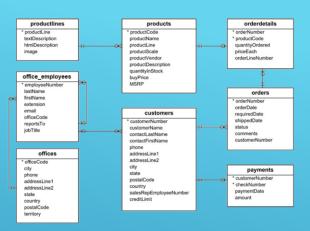


Return all customers who always made payments greater than 3000\$. Schema details can be found here.

SQL

Data Manipulation Language (DML)

Nested Queries (NOT IN)



Return all customers who always made payments greater than 3000\$. Schema details can be found here.



```
SELECT customerNumber, customerName FROM customers
WHERE customerNumber NOT IN (SELECT customerNumber FROM payments WHERE amount<=3000)
AND customerNumber IN (SELECT customerNumber FROM payments WHERE amount>3000)
```

What can go bad with such queries (NOT IN)?

productCode

productName

productVendor

customerName

addressLine1 addressLine2

state

postalCode

creditLimit

country

contactLastName

contactFirstName

productDescription quantityInStock

customers

salesRepEmployeeNumber

productLine

MSRP

productlines productLine

office\_employees

offices

textDescription

htmlDescription

image

lastName

firstName extension

officeCode

reportsTo

jobTitle

phone

state

country postalCode territory

addressLine1

addressLine2

email

(oneta@vaniercollege.qc.ca)

productCode

quantityOrdered priceEach

orderLineNumbe

orderDate

status

requiredDate

shippedDate

comments

orders

payments

\* checkNumber

paymentDate

SQL

Data Manipulation Language (DML)

Nested Queries (NOT IN)

Using below consultants table from here, return first name and last name for all office\_employees that

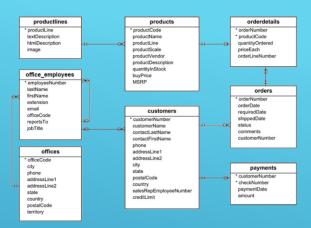
are not consultants. Schema details can be found here.

ast name for all office_employees that		
employeeNumber	vendorEmail	
1102	gbondur@vendors.com	
1337	lbondur@vendors.com	
1611	afixter@vendors.com	
1625	vkato@vendors.com	

SQL

Data Manipulation Language (DML)

Nested Queries (NOT IN)



Using below consultants table from <a href="here">here</a>, return first name and last name for all office\_employees that

are not consultants. Schema details can be found here.

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

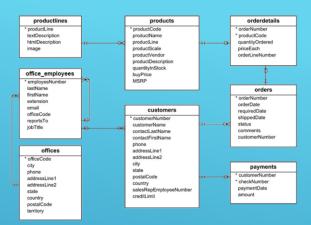


```
SELECT firstName, lastName FROM office_employees
WHERE employeeNumber NOT IN (SELECT employeeNumber FROM consultants);
```

SQL

Data Manipulation Language (DML)

Nested Queries (NOT IN)



Using below consultants table from <a href="here">here</a>, return first name and last name for all office\_employees that

are not consultants. Schema details can be found here.

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



```
SELECT firstName, lastName FROM office employees
WHERE employeeNumber NOT IN (SELECT employeeNumber FROM consultants);
```

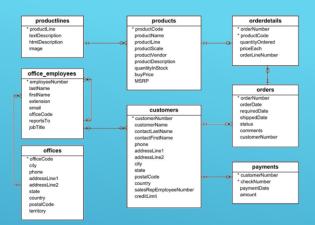
Let us now add a new vendor employee for which we only know his email address:

```
INSERT INTO consultants (employeeNumber, vendorEmail) VALUES (NULL, 'abt@vendors.com');
```

SQL

Data Manipulation Language (DML)

Nested Queries (NOT IN)



Using below consultants table from <a href="here">here</a>, return first name and last name for all office\_employees that

are not consultants. Schema details can be found here.

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



SELECT firstName, lastName FROM office\_employees
WHERE employeeNumber NOT IN (SELECT employeeNumber FROM consultants);



Is this the correct result?

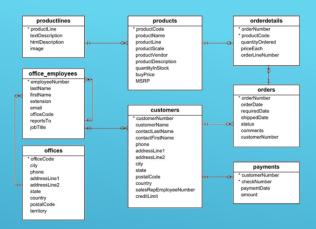
firstName

**lastName** 

SQL

Data Manipulation Language (DML)

Nested Queries (column operator ANY|SOME|ALL)

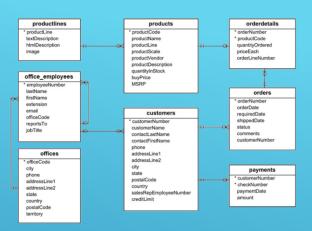


Return customer names for those costumers which has at least one payment greater than 3000. Schema details can be found <u>here</u>.

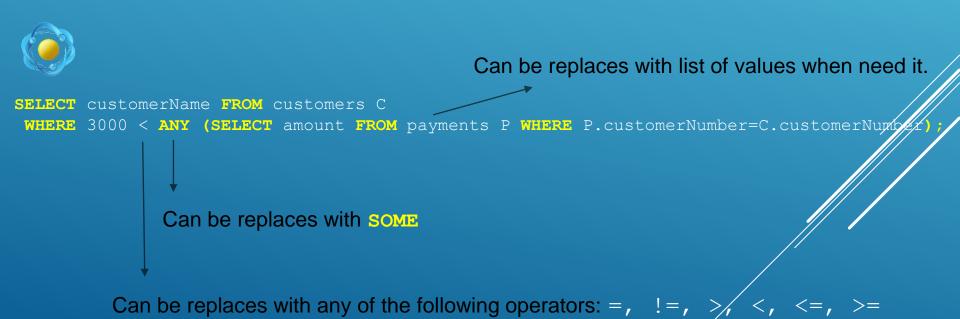
SQL

Data Manipulation Language (DML)

Nested Queries (column operator ANY|SOME|ALL)



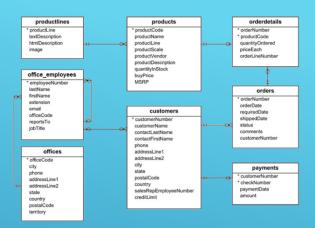
Return customer names for those costumers which has at least one payment greater than 3000. Schema details can be found <u>here</u>.



SQL

Data Manipulation Language (DML)

Nested Queries (column operator ANY|SOME|ALL)



Return customer names for those costumers which has all payments greater than 3000. Schema details can be found <u>here</u>.



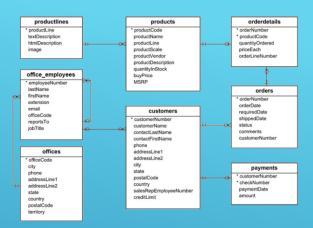
```
SELECT customerName FROM customers C
WHERE 3000 < ALL (SELECT amount FROM payments P WHERE P.customerNumber=C.customerNumber)
```

Is this correct?

SQL

Data Manipulation Language (DML)

Nested Queries (column operator ANY|SOME|ALL)



Return customer names for those costumers which has all payments greater than 3000. Schema details can be found here.



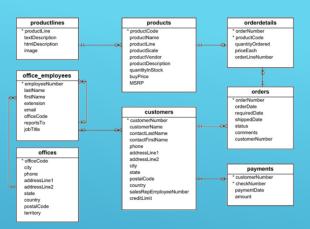
Lets check for customerNumber 247.

```
SELECT customerName FROM customers C
 WHERE 3000 < ALL (SELECT amount FROM payments P WHERE P.customerNumber=C.customerNumber
AND customerNumber=247;
                                               customerNumber
                                               247
```

SQL

Data Manipulation Language (DML)

Nested Queries (column operator ANY|SOME|ALL)



Return customer names for those costumers which has all payments greater than 3000. Schema details can be found here.



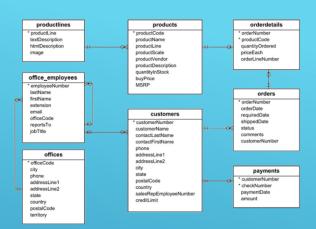
```
SELECT customerName FROM customers C
 WHERE 3000 < ALL (SELECT amount FROM payments P WHERE P.customerNumber=C.customerNumber
AND customerNumber=247;
                                               customerNumber
                                               247
Let us now see the payments for this customer.
SELECT amount FROM payments P WHERE P.customerNumber=247
                                                    amount
```

Is this what we want?

SQL

Data Manipulation Language (DML)

Nested Queries (column operator ANY|SOME|ALL)



We can rewrite **ALL** using **ANY** expression as follows:



```
SELECT customerName FROM customers C
WHERE 3000 < ALL (SELECT amount FROM payments P WHERE P.customerNumber=C.customerNumber)</pre>
```



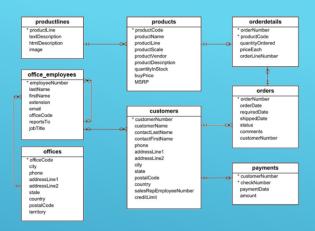
```
SELECT customerName FROM customers C WHERE
NOT (3000 >= ANY (SELECT amount FROM payments P WHERE P.customerNumber=C.customerNumber));
```

Boolean NOT operator.

SQL

Data Manipulation Language (DML)

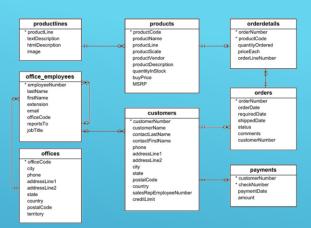
Nested Queries (EXISTS)



Return customer names for those costumers which has at least one payment greater than 3000 without using JOIN or ANY. Schema details can be found here.

Data Manipulation Language (DML)

Nested Queries (EXISTS)



Return customer names for those costumers which has at least one payment greater than 3000. Schema details can be found <u>here</u>.



```
SELECT customerName FROM customers C WHERE

EXISTS (SELECT amount FROM payments P WHERE P.customerNumber=C.customerNumber

AND payment > 3000);

SELECT customerName FROM customers C

WHERE 3000 < ANY (SELECT amount FROM payments P WHERE P.customerNumber=C.customerNumber);

SELECT DISTINT customerName FROM customers C INNER JOIN payments P

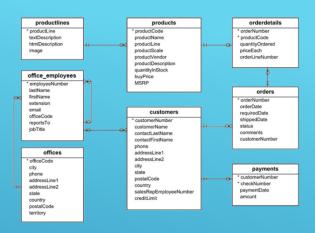
ON P.customerNumber=C.customerNumber AND P.payment > 3000;
```

## SQL

Data Manipulation Language (DML)

Nested Queries (EXISTS)

Consider consultans table



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

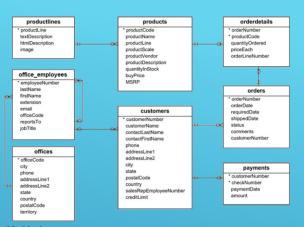


```
CREATE TABLE consultants (employeeNumber INT NULL, vendorEmail VARCHAR (100) NULL)
INSERT INTO consultants (employeeNumber, vendorEmail) VALUES (1102, 'gbondur@vendo/s.com');
INSERT INTO consultants (employeeNumber, vendorEmail) VALUES (1337, 'lbondur@vendors.com');
INSERT INTO consultants (employeeNumber, vendorEmail) VALUES (1611, 'afixter@vendors.com');
INSERT INTO consultants (employeeNumber, vendorEmail) VALUES (1625, 'ykato@vendors.com');
INSERT INTO consultants (employeeNumber, vendorEmail) VALUES (NULL, 'abt@vendors.com');
```

SQL

Data Manipulation Language (DML)

Nested Queries (EXISTS)



Note that **EXISTS** evaluates to **True** even if the returned column is NULL.

Using consultants table, return 1 if exists and employee with the vendor email 'abt@vendors.com'.

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



EXISTS (SELECT employeeNumber FROM consultants WHERE vendorEmail='abt@vendors.com









SELECT 1 FROM DUAL WHERE
EXISTS (SELECT employeeNumber

EXISTS (SELECT employeeNumber FROM consultants WHERE vendorEmail='abt@vendors.com');



### Data Manipulation Language (DML)

Nested Queries (implementing DIVISION) STUDENTS

### BORROWED

#### **BOOKS**

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

student_id	book_id
1	id100
1	id200
3	id200
1	ld206

id200

3

```
book_idauthortitleid100UllmanDBMSid200LinzAutomataid206BaaderTerm Rew.
```

```
CREATE TABLE Students(student_id INT, name VARCHAR(20), gender CHAR(1));

CREATE TABLE Borrowed(student_id INT, book_id VARCHAR(20));

CREATE TABLE Books(book_id VARCHAR(20), author VARCHAR(20), title VARCHAR(20));

INSERT INTO Students(student_id, name, gender) VALUES (1, 'John', 'M');

INSERT INTO Students(student_id, name, gender) VALUES (2, 'Adam', 'M');

INSERT INTO Students(student_id, name, gender) VALUES (3, 'Sandra', 'F');

INSERT INTO Books(book_id, author, title) VALUES ('id100', 'Ullman', 'DBM9');

INSERT INTO Books(book_id, author, title) VALUES ('id200', 'Linz', 'Authorata');

INSERT INTO Borrowed(student_id, book_id) VALUES (1, 'id200');

INSERT INTO Borrowed(student_id, book_id) VALUES (3, 'id200');

INSERT INTO Borrowed(student_id, book_id) VALUES (1, 'id206');

INSERT INTO Borrowed(student_id, book_id) VALUES (1, 'id200');

INSERT INTO Borrowed(student_id, book_id) VALUES (3, 'id200');
```

## Data Manipulation Language (DML)

Nested Queries (implementing DIVISION)

A-1		ч-	TA
		A I	
		F. I	

### **BORROWED**

### **BOOKS**

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

student_id	book_id
1	id100
1	id200
3	id200
1	ld206
3	id200

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

Find all student names that borrowed all books.

## Data Manipulation Language (DML)

Nested Queries (implementing DIVISION)

**STUDENTS** 

**BORROWED** 

**BOOKS** 

name	gender
John	М
Adam	М
Sandra	F
	John Adam

student_id	book_id
1	id100
1	id200
3	id200
1	ld206
3	id200

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

Find all student names that borrowed all books.

Step 1. Find all possible pairs of student\_id and book\_id.

## Data Manipulation Language (DML)

Nested Queries (implementing DIVISION)

**STUDENTS** 

name

John

Adam

Sandra

**BORROWED** 

id200

**BOOKS** 

gender	student_id	book_id
М	1	id100
М	Ī	id200
F	3	id200
	1	ld206

3

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

Find all student names that borrowed all books.

Step 1. Find all possible pairs of student\_id and book\_id.

3

student\_id



SELECT S.student\_id, B.books\_id FROM students S, books B;

student_id	book_id
1	id100
1	id200
1	id206
2	id100
2	id200
2	id206
3	id100
3	id200
3	id206

### Data Manipulation Language (DML)

Nested Queries (implementing DIVISION)

#### **STUDENTS**

#### **BORROWED**

#### **BOOKS**

student_id	name	gender	student_id	book_i
1	John	М	1	id100
2	Adam	М	1	id200
3	Sandra	F	3	id200
			1	ld206

DOOK_IG	aumor	ille
id100	Ullman	DBMS

id200 Linz **Automata** id206 Baader Term Rew.

Find all student names that borrowed all books.

### Step 2. Find all student ids that **did not** borrowed all the books.





S.student id, B.books id FROM students S, books B

EXCEPT

SELECT S.student id, B.books id FROM borrowed;



id200

student_id	book_id
2	id100
2	id200
2	id206
3	id100
3	id206



(SELECT S.student id, B.book id FROM students S, books B)

LEFT JOIN borrowed BO ON A.student id=BO.student id AND A.book id=BO.book id

3

WHERE BO.student id IS NULL AND BO.book id IS NULL;

## Data Manipulation Language (DML)

Nested Queries (implementing DIVISION)

CT			A I	
			M	
$\mathbf{v}$	$\mathbf{U}$		N	

#### **BORROWED**

#### **BOOKS**

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

student_id	book_id	
1	id100	
1	id200	
3	id200	
1	ld206	

id200

3

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

Find all student names that borrowed all books.

### Step 3. Find all students that **did** borrowed all the books.



);





### Data Manipulation Language (DML)

Nested Queries (implementing DIVISION)

CT		- 10	l bu	(3)
21	U	- 1	Ш	$\mathbf{a}$

#### **BORROWED**

#### **BOOKS**

name	gender
John	М
Adam	М
Sandra	F
	John Adam

student_id	book_id		
1	id100		
1	id200		
3	id200		
1	ld206		
3	id200		

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

Find all student names that borrowed all books.

### Step 3. Find all students that **did** borrowed all the books.



);

```
SELECT name FROM students
WHERE student id NOT IN
(SELECT A.student id
   FROM
         (SELECT S.student id, B.book id FROM students S, books B) A
          LEFT JOIN borrowed BO ON A.student id=BO.student id AND A.book id=BO.book id
  WHERE BO.student id IS NULL AND BO.book id IS NULL
```

Name adam

### Data Manipulation Language (DML)

CASE expression

```
CASE input_expression
     WHEN when_expression THEN result_expression [ ...n ]
     [ ELSE else_result_expression ]
END;
```

Evaluates to the first result\_expression if input\_expression is equal to the when\_expression if not else\_result\_expression is returned.

```
WHEN boolean_expression THEN result_expression [ ...n ]
   [ ELSE else_result_expression ]
END;
```

Evaluates to the first result\_expression where the boolean\_expression is returned.

Data Manipulation Language (DML)

CASE expression

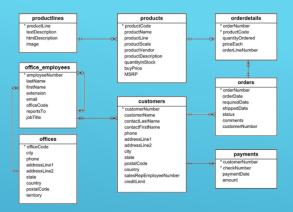
Return name of the product and product line mapped as below for all products with quantity in stock greater then 3000. Schema details can be found <a href="https://example.com/here">https://example.com/here</a>. Product Line Name To Be Displayed

product	ines	products		orderdetails
* productLine		* productCode	<b>-</b>	* orderNumber
textDescription	in	productName		* productCode
htmlDescripti	on H	→ productLine	H OH	quantityOrdered
image		productScale		priceEach
		productVendor		orderLineNumber
		productDescription		
		quantityInStock		<b>*</b>
office_emp		buyPrice		Ϋ́
* employeeNu	mber	MSRP		#
lastName			ا لـ	orders
firstName				* orderNumber
extension				orderDate
email		customers	7	requiredDate
officeCode		* customerNumber	-	shippedDate
reportsTo	<del>  10</del> J	customerName	++∞<	status
jobTitle	10	customername	11 ~~~	comments
		contactCastName		customerNumber
		phone		Customentumber
office		phone addressLine1	L L	
* officeCode	3	addressLine2		
city		city	Г	payments
phone		state		
addressLine1		postalCode	H OH	* customerNumber
addressLine2		country		* checkNumber
state		salesRepEmployeeNumber		paymentDate
country		creditLimit		amount
postalCode				

Product Line	Name To Be Displayed
Classic Cars	Classic
Motorcycles	Bikes
Planes	Planes
Ships	Ships
Trains	Trains
Trucks and Buses	Large
Vintage Cars	Vintage

Data Manipulation Language (DML)

CASE expression



Return name of the product and product line mapped as below for all products with quantity in stock greater then 3000. Schema details can be found here.

Product Line Name To Be Displayed

```
CASE productLine

WHEN 'Classic Cars' THEN 'Classic'

WHEN 'Motorcycles' THEN 'Bikes'

WHEN 'Trucks and Buses' THEN 'Large'

WHEN 'Vintage Cars' THEN 'Vintage'

ELSE productLine

END AS prodGroup

FROM products

WHERE quantityInStock>3000;
```

Classic Cars
Classic
Motorcycles
Bikes
Planes
Planes
Ships
Ships
Trains
Trucks and Buses
Large
Vintage Cars
Name To Be Displayed
Classic
Notation To Be Displayed
Lassic
Name To Be Displayed
Lassic
Name To Be Displayed
Lassic
Name To Be Displayed
Vintage
Lassic
Vintage
Vintage
Vintage

SQL

Data Manipulation Language (DML)

**CASE** expression

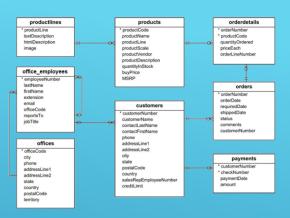
productLine textDescriptio htmlDescription productLine quantityOrdered productVendor orderl ineNumb productDescription quantityInStock office\_employees MSRP orderDate requiredDate officeCode shippedDate customerName status iobTitle contactLastName contactFirstName customerNumbe addressl ine1 officeCode addressLine2 postalCode addressLine1 checkNumber country salesRepEmploy paymentDate state creditLimit postalCode

Return name of the product and a column that displays either "Large" if the scale is larger or equal to 1:12, "Medium" if scale is strictly smaller than 1:12 and larger or equal to 1:24 and "Small" otherwise. Schema details can be found <a href="https://example.com/here">here</a>.

## SQL

Data Manipulation Language (DML)

CASE expression



Return name of the product and a column that displays either "Large" if the scale is larger or equal to 1:12, "Medium" if scale is strictly smaller than 1:12 and larger or equal to 1:24 and "Small" otherwise. Schema details can be found here.



```
CASE

WHEN CAST (REPLACE (productScale, '1:','') AS INT) <= 12 THEN 'LARGE'
WHEN CAST (REPLACE (productScale, '1:','') AS INT) <= 24 THEN 'MEDIUM'
ELSE 'SMALL'
END AS prodSize
FROM products;

AS UNASIGNED
```