SQL

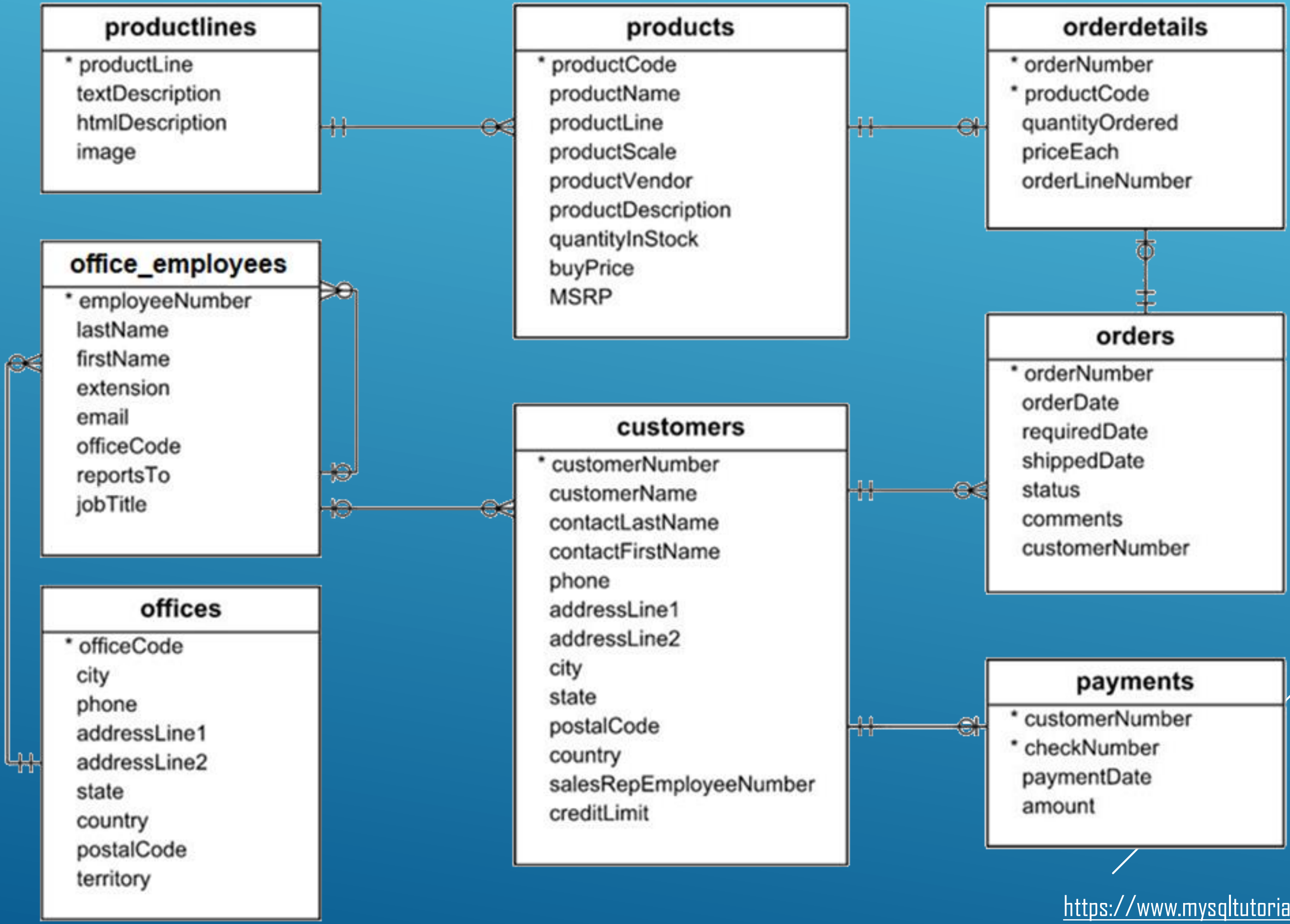
PART II (SELECT)

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)

Select Syntax

```
SELECT list_of_fields|expressions
   [ FROM list_of_tables
     [ WHERE condition ]
     [ GROUP BY list_of_fields ]
     [ HAVING condition ]
   [ ORDER BY list_of_fields ]]
[UNION [ALL]
SELECT …     ]
```

SQL

Data Manipulation Language (DML)

Select without tables

```
SELECT 1 as MyNumber
```

MyNumber
1

```
SELECT 1 as MyNumber FROM dual
```

MySQL accepts both ways.

SQL

Data Manipulation Language (DML)

Select without tables

Note that we may add even where condition in both cases (with **`DUAL`** or without **`FROM`**).

SQL

Data Manipulation Language (DML)

Projection (π) Which columns/expressions to be returned.

STUDENTS

student_id	name	gender
1	John	M
2	Mike	M
3	Marry	F

SQL

Data Manipulation Language (DML)

Projection (π) Returning all columns.

STUDENTS

student_id	name	gender
1	John	M
2	Mike	M
3	Marry	F

```
SELECT * FROM students
```

student_id	name	gender
1	John	M
2	Mike	M
3	Marry	F

Except Oracle

```
SELECT *, name FROM students
```

student_id	name	gender	name
1	John	M	John
2	Mike	M	Mike
3	Marry	F	Marry

SQL

Data Manipulation Language (DML)

Projection (π) Returning specific columns.

STUDENTS

student_id	name	gender
1	John	M
2	Mike	M
3	Marry	F

```
SELECT student_id FROM students
```

student_id
1
2
3

```
SELECT gender FROM students
```

gender
M
M
F

SQL

Data Manipulation Language (DML)

Projection (π) Eliminate duplication.

Return existing genders for the students (π_{gender} (students)).

STUDENTS

student_id	name	gender
1	John	M
2	Mike	M
3	Marry	F

```
SELECT gender FROM students
```

gender
M
M
F

```
SELECT DISTINCT gender FROM students
```

gender
M
F

SQL

Data Manipulation Language (DML)

Projection (π) Adding expressions (string concatenation, column alias).

STUDENTS

student_id	name	gender
1	John	M
2	Mike	M
3	Marry	F

For each student return a string containing student's name and his/her gender in parenthesis.

```
SELECT name + ' (' + gender + ')' AS sg FROM students
```

```
SELECT name || ' (' || gender || ')' AS sg FROM students
```

```
SELECT name & ' (' & gender & ')' AS sg FROM students
```

```
SELECT CONCAT(name,CONCAT(' (',CONCAT(gender,')'))) AS sg FROM students
```

```
SELECT CONCAT(name,' (',gender,')') AS sg FROM students
```

```
SELECT name CONCAT ' (' CONCAT gender CONCAT ')'  AS sg FROM students
```

sg
John (M)
Mike (M)
Marry (F)

SQL

Data Manipulation Language (DML)

Select a limited number or rows

STUDENTS

student_id	name	gender
1	John	M
2	Mike	M
3	Marry	F

Return only 2 students.

Access

```
SELECT TOP 2 * FROM students
```

```
SELECT * FROM students LIMIT 2
```

```
SELECT * FROM students WHERE rownum<=2
```

```
SELECT * FROM students FETCH FIRST 2 ROWS ONLY
```

student_id	name	gender
1	John	M
2	Mike	M

SQL

Data Manipulation Language (DML)

Selection (σ)

Return all male students ($\sigma_{gender='M'}$ (students)).

```
SELECT * FROM students WHERE gender='M'
```

STUDENTS

student_id	name	gender
1	John	M
2	Mike	M
3	Marry	F

student_id	name	gender
1	John	M
2	Mike	M

SQL

Data Manipulation Language (DML)

Selection and Projection (σ)

STUDENTS

student_id	name	gender
1	John	M
2	Mike	M
3	Marry	F

Return student Id and name for all male students ($\pi_{\text{student\_id, name}}(\sigma_{\text{gender='M'}}(\text{students}))$).

```
SELECT student_id, name FROM students WHERE gender='M'
```

student_id	name	gender
1	John	M
2	Mike	M

SQL

Data Manipulation Language (DML)

Selection (σ). String pattern matching

STUDENTS

student_id	name	gender
1	John	M
2	Mike	M
3	Marry	F

Return student Id and name for all students which name starts with letter M.

```
SELECT student_id, name FROM students
 WHERE name LIKE 'M%'
```

```
% - any string
_ - single character
```

student_id	name	gender
2	Mike	M
3	Marry	F

Access

```
SELECT student_id, name FROM students
 WHERE name LIKE 'M*'
```

```
* - any string
? - single character
```

For complementary condition you may use **NOT LIKE**

SQL

Data Manipulation Language (DML)

Selection (σ). String pattern matching with regular expression

Return the name for all students which name ends with n.

STUDENTS

student_id	name	gender
1	John	M
2	Mike	M
3	Marry	F

```
SELECT name FROM students WHERE REGEXP_LIKE(name,'n$')
```

name
John

```
SELECT name FROM students WHERE REGEXP_MATCH(name,'n$','g')='n'
```

SQL

Data Manipulation Language (DML)

Selection (σ). Date/Time formatting

In MySQL default date format is in format 'YYY-MM-DD'.

```
UPDATE orders SET orderDate='2020-11-27'
WHERE orderDate=NOW();
```

You can convert date into a string using **DATE_FORMAT** function.

```
UPDATE orders SET
orderDescription=DATE_FORMAT(orderdate, '%m-%d-%Y')
WHERE orderDate=NOW();
```

You can convert a string into a date using **STR_TO_DATE** function.

More formatting styles can be found: https://www.mysqltutorial.org/mysql-date_format

SQL

Data Manipulation Language (DML)

Selection (σ). Date/Time formatting

In Oracle you have to use TO_DATE function to convert strings to DATE.

```
TO_DATE(string_to_convert,format)
```

```
UPDATE orders
    SET orderDate=TO_DATE('2020-11-27','YYYY-MM-DD')
WHERE orderDate=CURRENT_DATE;
```

You can convert a string into a date using **TO_CHAR** function.

Format	Description
YYYY	4-digit year
YY	2-digit year
Q	Quarter of the year (1-4)
MON	Abbreviated month (Jan - Dec)
MONTH	Month name (January - December)
MM	Month (1 - 12)
IW	Week of the year (1-53)
DY	Abbreviated day (Sun - Sat)
DDD	Day of the year (1-366)
DD	Day of the month (1 - 31)
D	Day of the week (1-7)
DAY	Full name of the day
DY	Abbreviated name of the day
HH24	Hour (0 - 23)
HH or HH12	Hour (1 - 12)
MI	Minutes (0 - 59)
SS	Seconds (0 - 59)
SSSSS	Seconds past midnight (0-86399)
AM	Meridian indicator

SQL

Data Manipulation Language (DML)

Selection (σ). Date/Time formatting

```
UPDATE orders SET orderDate=CONVERT(DATE,'1/6/2003',101)
WHERE orderDate=getdate();
```

Formatting codes.
Formatting codes.

http://infocenter-archive.sybase.com/help/index.jsp?topic=/com.sybase.help.ase_15.0.blocks/html/blocks/blocks125.htm
https://docs.microsoft.com/en-us/sql/t-sql/functions/cast-and-convert-transact-sql?view=sql-server-ver15

Formatting codes.

You can convert a string into a date using `CONVERT(VARCHAR(20),date_value,101)` function.

SQL

Data Manipulation Language (DML)

NULLS

NULLS are placeholders for unknown/non-applicable values.
Database Management Systems uses 3-valued logic (True, False, Unknown).

Truth table OR

∨	**True**	**False**	**Unknown**
True	True	True	True
False	True	False	Unknown
Unknown	True	Unknown	Unknown

Truth table AND

∧	**True**	**False**	**Unknown**
True	True	False	Unknown
False	False	False	False
Unknown	Unknown	False	Unknown

Truth table NOT

	¬
True	False
False	True
Unknown	Unknown

SQL

Data Manipulation Language (DML)

NULLS. Evaluation

Expression evaluation under ANSI NULL

SQL Server

```
SET ANSI_NULLS {ON|OFF}
```

SYBASE

```
SET ANSINULL {ON|OFF}
```

ANSI NULL is always ON

PostgreSQL

You can test expression value:

```
expression IS UNKNOWN
```

Expression	ANSI NULL ON	ANSI NULL OFF
NULL = NULL	Unknown	True
1 = NULL	Unknown	False
NULL <> NULL	Unknown	False
1 <> NULL	Unknown	True
NULL > NULL	Unknown	Unknown
1 > NULL	Unknown	Unknown
NULL IS NULL	True	True
1 IS NULL	False	False
NULL IS NOT NULL	False	False
1 IS NOT NULL	True	True

SQL

Data Manipulation Language (DML)

NULLS. Evaluation

Offices

name	city
3Ab	NULL
9Cd	Berlin

What will return the following queries?

Query

```
SELECT * FROM Offices WHERE city<>"Berlin"

SELECT * FROM Offices WHERE city IS NOT NULL

SELECT * FROM Offices WHERE city = NULL

SELECT * FROM Offices WHERE city IS NULL
```

SQL

Data Manipulation Language (DML)

NULLS. Evaluation

Offices

name	city
3Ab	NULL
9Cd	Berlin

What will return the following queries?

Query

```
SELECT * FROM Offices WHERE city<>"Berlin"
```

$\rightarrow$

Name	City

```
SELECT * FROM Offices WHERE city IS NOT NULL

SELECT * FROM Offices WHERE city = NULL

SELECT * FROM Offices WHERE city IS NULL
```

SQL

Data Manipulation Language (DML)

NULLS. Evaluation

Offices

name	city
3Ab	NULL
9Cd	Berlin

What will return the following queries?

Query

```
SELECT * FROM Offices WHERE city<>"Berlin"

SELECT * FROM Offices WHERE city IS NOT NULL

SELECT * FROM Offices WHERE city = NULL

SELECT * FROM Offices WHERE city IS NULL
```

$\rightarrow$

Name	City
9Cd	Berlin

SQL

Data Manipulation Language (DML)

NULLS. Evaluation

Offices

name	city
3Ab	NULL
9Cd	Berlin

What will return the following queries?

Query

```
SELECT * FROM Offices WHERE city<>"Berlin"

SELECT * FROM Offices WHERE city IS NOT NULL

SELECT * FROM Offices WHERE city = NULL

SELECT * FROM Offices WHERE city IS NULL
```

→

Name	City

SQL

Data Manipulation Language (DML)

NULLS. Evaluation

Offices

name	city
3Ab	NULL
9Cd	Berlin

What will return the following queries?

Query

```
SELECT * FROM Offices WHERE city<>"Berlin"

SELECT * FROM Offices WHERE city IS NOT NULL

SELECT * FROM Offices WHERE city = NULL

SELECT * FROM Offices WHERE city IS NULL
```

$\rightarrow$

Name	City
3Ab	NULL

SQL

Data Manipulation Language (DML)

NULLS. COALESCE expression

Offices

name	city
3Ab	NULL
9Cd	Berlin

ISNULL(value1,value2) function returns value1 if not null otherwise returns value2.

SAP Sybase, SQL Server

```
SELECT * FROM Offices WHERE ISNULL(city,"Berlin") = "Berlin"
```

IBM DB2, MySQL

```
SELECT * FROM Offices WHERE IFNULL(city,"Berlin") = "Berlin"
```

Oracle

```
SELECT * FROM Offices WHERE NVL(city,"Berlin") = "Berlin"
```

Access

```
SELECT * FROM Offices WHERE IFF(city IS NULL,"Berlin", city) = "Berlin"
```

PostgreSQL, IBM DB2, Oracle, SAP Sybase, SQL Server

```
SELECT * FROM Offices WHERE COALESCE(city,"Berlin") = "Berlin"
```

Accepts multiple arguments and returns the first not null value.

SQL

Data Manipulation Language (DML)

NULLS. Semantics

Employees

Name	City
John	Montreal

Offices

Name	City
3Ab	NULL

Return all the employee office names for which the employee is in a different city or the office city is not New York.

Schema/Data Creation

```
CREATE TABLE employees (name VARCHAR(20), city VARCHAR(20) NULL);
CREATE TABLE offices (name VARCHAR(20), city VARCHAR(20) NULL);

INSERT INTO employees (name, city) VALUES ('John', 'Montreal');
INSERT INTO offices (name, city) VALUES ('3Ab', NULL);
```

Query

```
SELECT E.name, O.name FROM employees E, offices O
WHERE E.city<>O.city OR O.city<>'New York';
```

WHAT IS THE EXPECTED RESULT?

SQL

Data Manipulation Language (DML)

NULLS. Semantics

Employees

Name	City
John	Montreal

Offices

Name	City
3Ab	NULL

Return all the employee office names for which the employee is in a different city or the office city is not New York.

Query

```
SELECT E.name AS ename, O.name AS oname FROM employees E, offices O
WHERE E.city<>O.city OR O.city<>'New York';
```

WHAT IS THE EXPECTED RESULT?

There are 2 options for the office city:
1. Office City is 'New York'
2. Office City is not 'New York'

ename	oname
John	3Ab

The actual result given by the DBMS is the empty set.

SQL

Data Manipulation Language (DML)

Union (∪).

STUDENTS

student_id	name	gender
1	Jake	M
2	Mike	M
3	Marry	F

From students returns all student names that either starts with M or contains letter K.

```
SELECT name FROM students WHERE name LIKE 'M%'
UNION ALL
SELECT name FROM students WHERE name LIKE '%k%'
```

$\rightarrow$

name
Jake
Mike
Mike
Marry

! Can be performed only between record sets with the same number of columns and columns from the same position have compatible types.

SQL

Data Manipulation Language (DML)

Union (∪).

STUDENTS

student_id	name	gender
1	Jake	M
2	Mike	M
3	Marry	F

From students returns all student names that either starts with M or contains letter K.

```
SELECT name FROM students WHERE name LIKE 'M%'
UNION
SELECT name FROM students WHERE name LIKE '%k%'
```

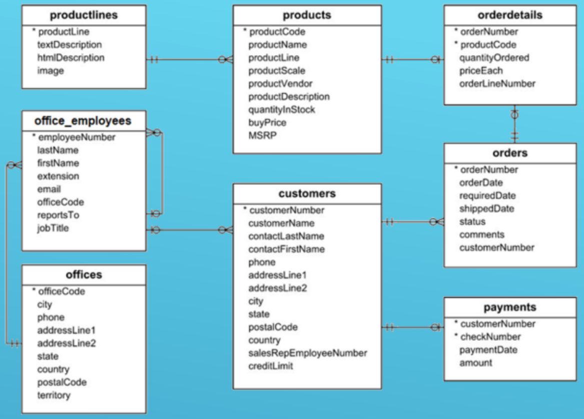
→

name
Jake
Mike
Marry

SQL

Data Manipulation Language (DML)

Difference (–).

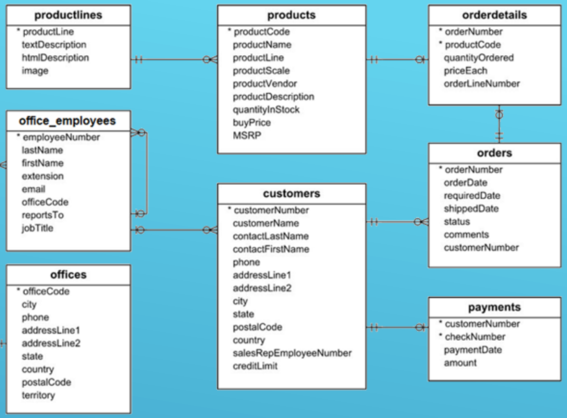


Return all product codes from productLine "Motorcycles" that were never ordered in a quantity greater than 50. Schema details can be found here.

SQL

Data Manipulation Language (DML)

Difference (–).



Return the codes for those "Motorcycles" (productLine) products that were never ordered in a quantity greater than 50. Schema details can be found here.

1. All products that are motorcycles.

```
SELECT productCode FROM products WHERE productLine='Motorcycles';
```

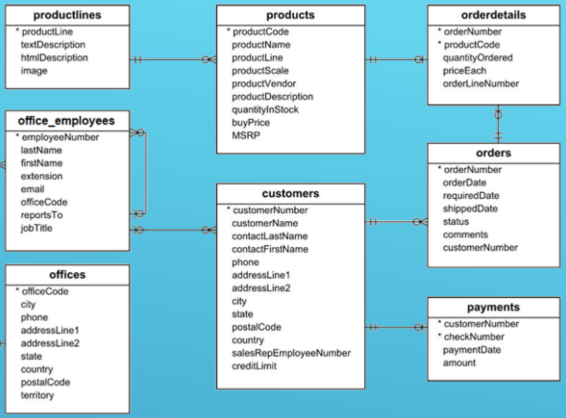
2. All products that were ordered in quantity greater than 50.

```
SELECT productCode FROM orderDetails WHERE quantityOrdered>50;
```

SQL

Data Manipulation Language (DML)

Difference (–).



Return the codes for those "Motorcycles" (productLine) products that were never ordered in a quantity greater than 50. Schema details can be found here.

! Can be performed only between record sets with the same number of columns and columns from the same position have compatible types.

Problem Solution

```
SELECT productCode FROM products WHERE productLine='Motorcycles'
EXCEPT
SELECT productCode FROM orderDetails WHERE quantityOrdered>50;
```

ORACLE

```
SELECT productCode FROM products WHERE productLine='Motorcycles'
MINUS
SELECT productCode FROM orderDetails WHERE quantityOrdered>50;
```

How about Access ? See solution here.

SQL

Data Manipulation Language (DML)

Difference (–).

Few points to mention:

1. Difference is considered as set difference.

Example:

A

Col
1
2
2
3
3

B

Col
1
2
4

```
SELECT col FROM A
EXCEPT
SELECT col FROM B;
```

↓

?

SQL

Data Manipulation Language (DML)

Difference (–).

Few points to mention:

1. Difference is considered as set difference.

Example:

A

Col
1
2
2
3
3

B

Col
1
2
4

```
SELECT col FROM A
EXCEPT
SELECT col FROM B;
```

Col
3

SQL

Data Manipulation Language (DML)

Difference (–).

Few points to mention:

2. For difference NULLs are considered equal.

Example:

A

Col
1
2
2
3
NULL

B

Col
1
2
4
NULL

```
SELECT col FROM A
EXCEPT
SELECT col FROM B;
```

↓

?

SQL

Data Manipulation Language (DML)

Difference (–).

Few points to mention:

2. For difference NULLs are considered equal.

Example:

A

Col
1
2
2
3
NULL

B

Col
1
2
4
NULL

```
SELECT col FROM A
EXCEPT
SELECT col FROM B;
```

↓

Col
3

SQL

Data Manipulation Language (DML)

Difference (–).

Few points to mention:

3. Returned record set will contain column naming given by the first select.

Example:

A

Col1
1
2
2
3
NULL

B

Col2
1
2
4
NULL

```
SELECT col1 FROM A
EXCEPT
SELECT col2 FROM B;
```

Col1
3

SQL

Data Manipulation Language (DML)

Difference (–).

Few points to mention:

4. Difference/Union can be performed on multiple tables.

Example:

A

Col1
1
2
2
3
NULL

B

Col2
1
2
4
NULL

C

Col3
3
5

```
SELECT col1 FROM A
EXCEPT
SELECT col2 FROM B
EXCEPT
SELECT col3 FROM C;
```

(A-B)-C

Col1

```
SELECT col1 FROM A
UNION
SELECT col3 FROM C
EXCEPT
SELECT col2 FROM B;
```

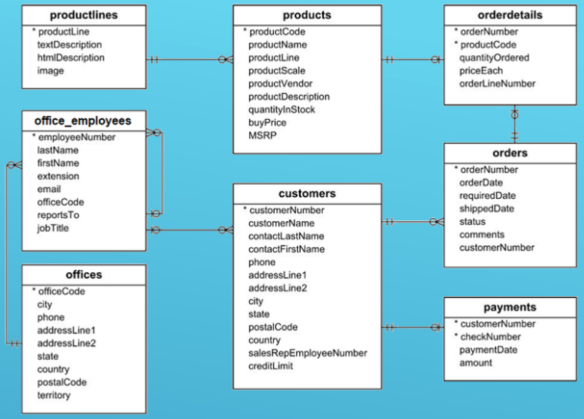
(A∪C)-B

Col1
3
5

SQL

Data Manipulation Language (DML)

Intersection (∩).

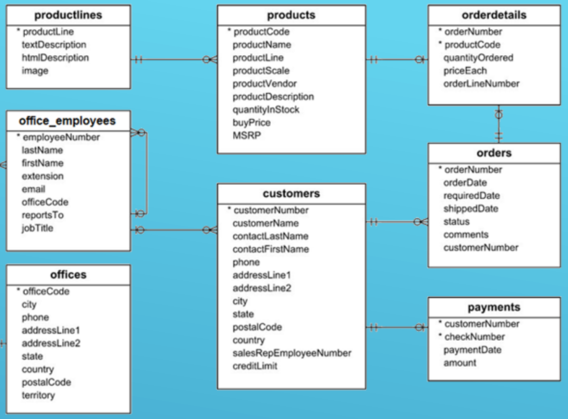


Return the codes for those "Motorcycles" (productLine) products that were ordered at least once in a quantity greater than 50. Schema details can be found here.

SQL

Data Manipulation Language (DML)

Intersection (∩).



Return the codes for those "Motorcycles" (productLine) products that were ordered at least once in a quantity greater than 50. Schema details can be found here.

! Can be performed only between record sets with the same number of columns and columns from the same position have compatible types.

Problem Solution

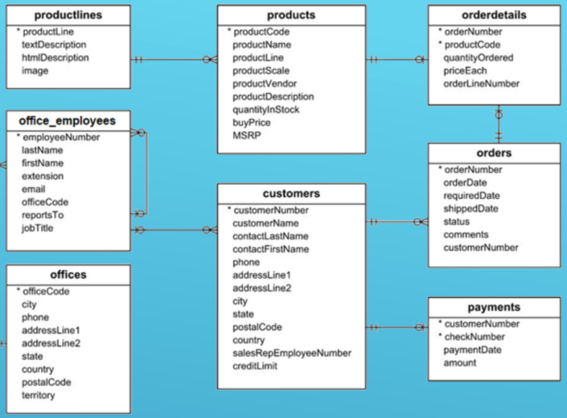
```
SELECT productCode FROM products WHERE productLine='Motorcycles'
INTERSECT
SELECT productCode FROM orderDetails WHERE quantityOrdered>50;
```

How about ?

SQL

Data Manipulation Language (DML)

Intersection (∩).



Return the codes for those "Motorcycles" (productLine) products that were ordered at least once in a quantity greater than 50. Schema details can be found here.

! Can be performed only between record sets with the same number of columns and columns from the same position have compatible types.

Problem Solution

```
SELECT productCode FROM products WHERE productLine='Motorcycles'
INTERSECT
SELECT productCode FROM orderDetails WHERE quantityOrdered>50;
```

How about SAP SYBASE MySQL Access ? Note that A∩B = A – (A – B)

As we will see we can also mimic intersection using joins, **IN** or **EXISTS** expressions.
(Homework – pay attention to duplication)