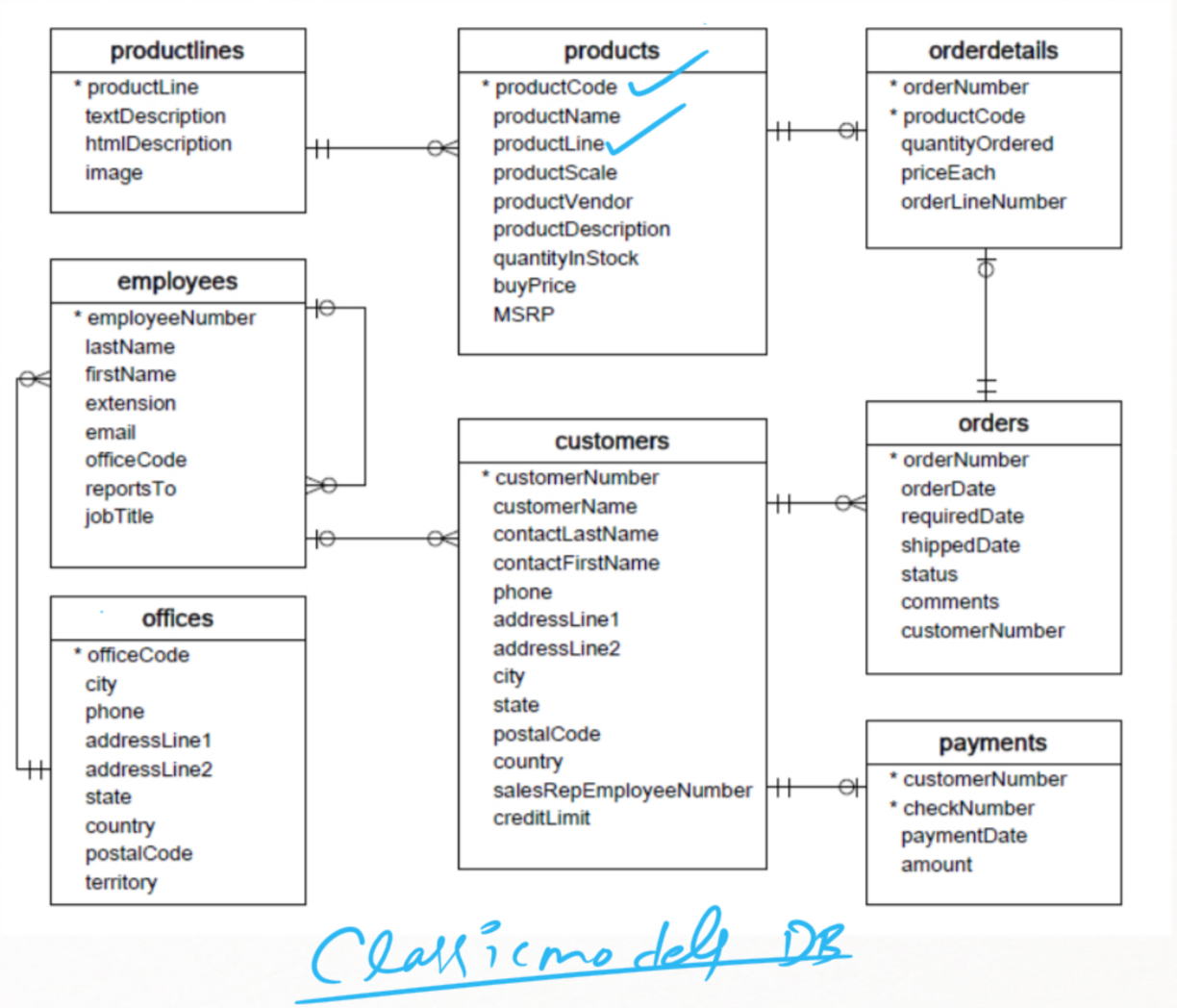
DQL : Data Query lang.

select, where, group By

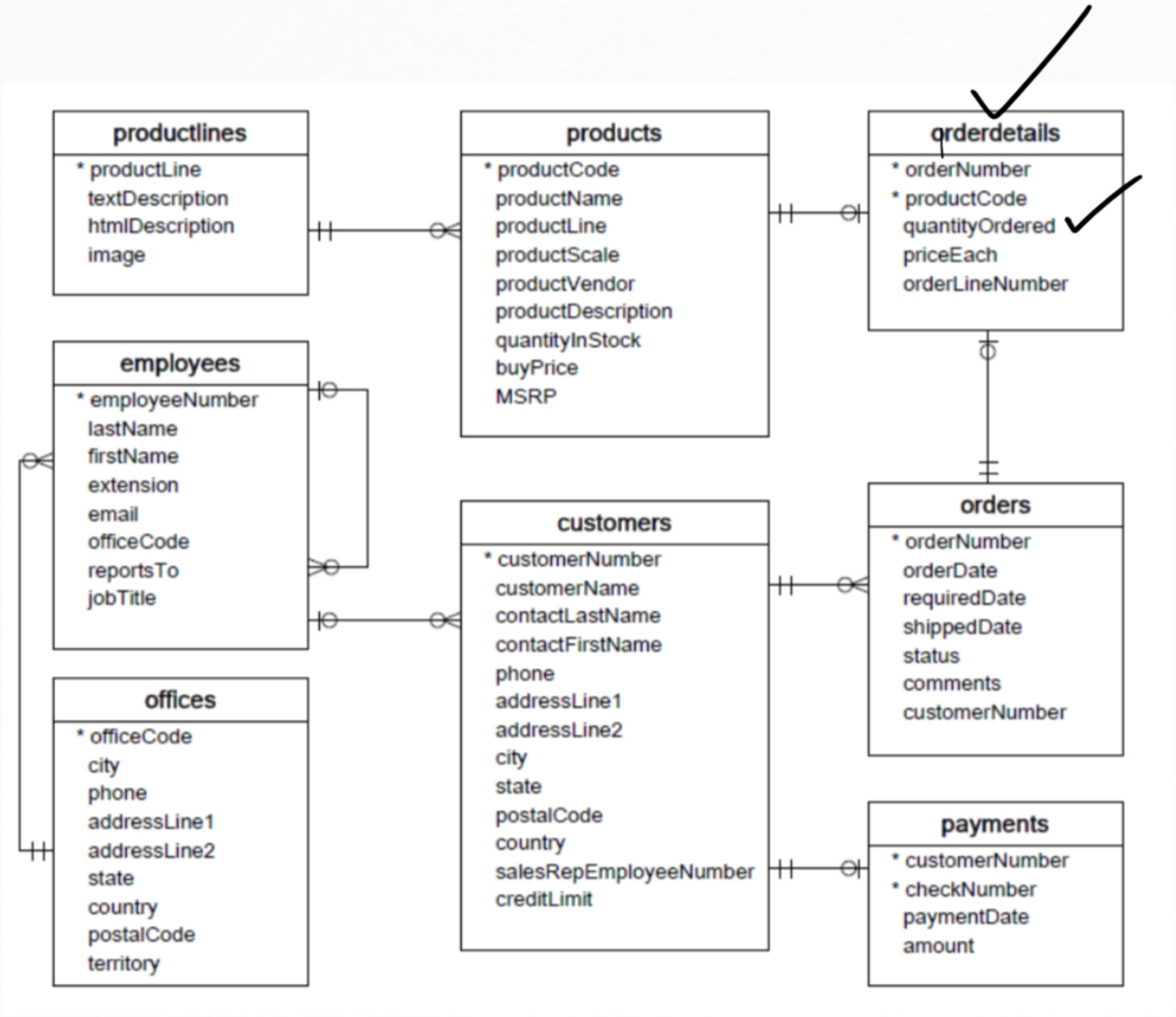
Having, order by, like

Q1. Print productline and product code for all products.

=> Select pl, pc from products.



Classicmodels DB



Q2. Print productlines & productCode but the colum name should be line and code respectively.

=) Select productline as line, productcode as code from products.

Q3. Print all the orders (of that product) whose quantity ordered more than 3.

3. =) Select * from orderdetails where quantityOrdered > 3

Name	Fees	Branch
A	50K	IT
B	35K	CS
C	70K	CS
D	80K	IT
E	100K	ECE

College

Branch	Avg(Fees)
IT	50+80/2
CS	105/2
ECE	100

select sum(st.id), major from st group by major.

Q1. Avg Fees of each branch.

Select AVG(Fees), Branch Feom College Group By Branch.

student_id	name	major
1	Jack	Biology
2	Kate	Sociology
3	Claire	English
4	Jack	Biology
5	Mike	Comp. Sci

Q2. Print all different majors and Sum of their Student_id's studying in that major.

St.

sum(st.id)	major
5	Bio
2	Soci
3	Eng
5	CompSci

Note =) * All the Aggregate functions when used with group by will be acting on individual groups which are formed by group by.

LIKE Operator

=) pattern matching.

=) (1) _ =) Single Char

(2) % =) 0 or more characters.

emp_id	first_name	last_name	birth_date	sex	salary	super_id	branch_id
100	David	Wallace	1967-11-17	M	250,000	NULL	1
101	Jan	Levinson	1961-05-11	F	110,000	100	1
102	Michael	Scott	1964-03-15	M	75,000	100	2
103	Angela	Martin	1971-06-25	F	63,000	102	2
104	Kelly	Kapoor	1980-02-05	F	55,000	102	2
105	Stanley	Hudson	1958-02-19	M	69,000	102	2
106	Josh	Porter	1969-09-05	M	78,000	100	3
107	Andy	Bernard	1973-07-22	M	65,000	106	3
108	Jim	Halpert	1978-10-01	M	71,000	106	3

a/o → o or more char

```
mysql> SELECT first_name,last_name
    -> FROM Employee
    -> WHERE last_name LIKE 'H%';
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| Stanley    | Hudson    |
| Jim        | Halpert   |
+------------+-----------+
2 rows in set (0.00 sec)
```

Q1. Find first_name of all employee whose last_name starts with H.

⇒ select first_name, last_name ✓✓
from emp
where last_name LIKE 'H%'

HAVING CLAUSE

→ Used with group By for conditions on Agg function results.

→ If Used without group By ≃ WHERE CLAUSE.

→ It is always utilised on groups.

→ WHERE condition and HAVING can be used in 1 Query.

Agg ↑ Sum, Count, Avg

WHERE condition works on the data before the groups are formed

where sum(salary) > 250k

Wrong.

HAVING sum(salary) > 250k

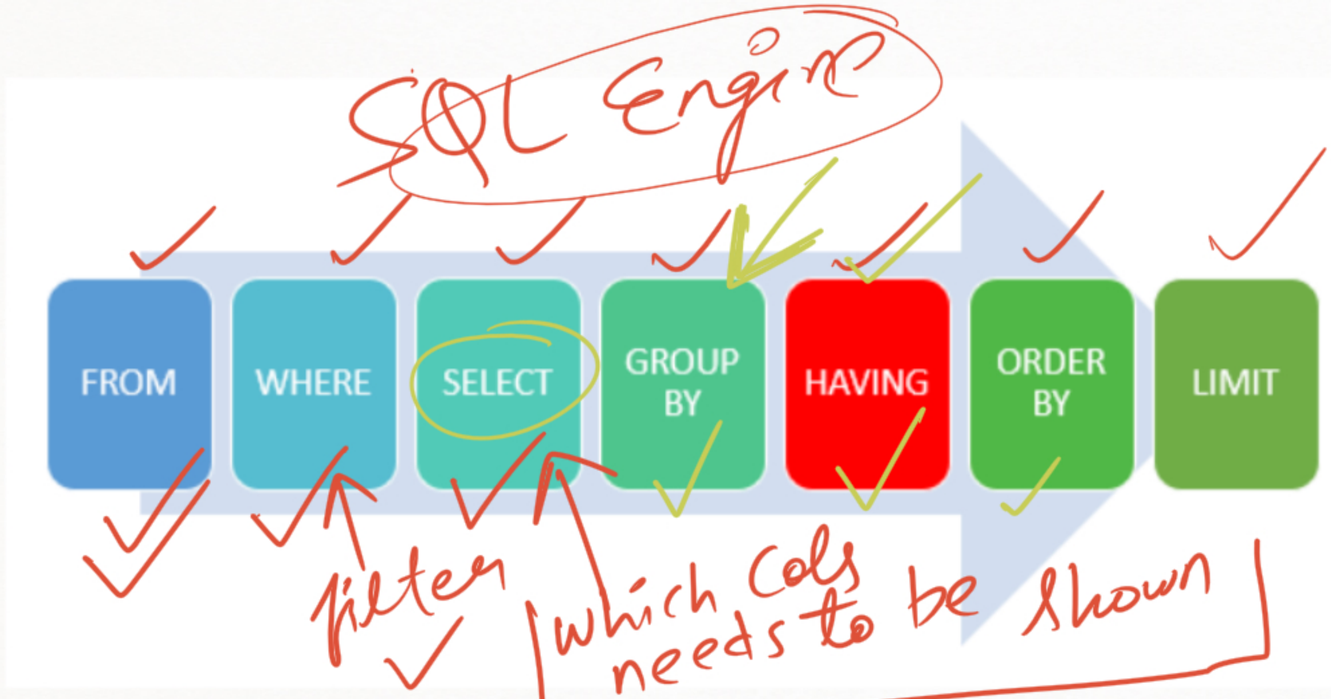
```
mysql> SELECT SUM(salary),branch_id
    -> FROM Employee
    -> GROUP BY branch_id
    -> HAVING SUM(salary) > 250000;
+-------------+-----------+
| SUM(salary) | branch_id |
+-------------+-----------+
|      360000 |         1 |
|      262000 |         2 |
+-------------+-----------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT SUM(salary),branch_id
    -> FROM Employee
    -> GROUP BY branch_id;
+-------------+-----------+
| SUM(salary) | branch_id |
+-------------+-----------+
|      360000 |         1 |
|      262000 |         2 |
|      214000 |         3 |
+-------------+-----------+
3 rows in set (0.00 sec)
```

HAVING CLAUSE ✓

```
mysql> SELECT SUM(salary),branch_id
    -> FROM Employee
    -> GROUP BY branch_id
    -> WHERE SUM(salary) > 250000;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual
on for the right syntax to use near 'WHERE SUM(salary) > 250000' at line 4
mysql> SELECT SUM(salary),branch_id
    -> FROM Employee
    -> WHERE SUM(salary) > 250000
    -> GROUP BY branch_id;
ERROR 1111 (HY000): Invalid use of group function
```

Errors when WHERE Used Instead Having.



Order in which SQL Keywords will be executed.

```
mysql> SELECT SUM(salary),branch_id
    -> FROM Employee
    -> GROUP BY branch_id
    -> HAVING SUM(salary) > 250000
    -> ORDER BY SUM(salary) ASC;
+-------------+-----------+
| SUM(salary) | branch_id |
+-------------+-----------+
|      262000 |         2 |
|      360000 |         1 |
+-------------+-----------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT SUM(salary),branch_id
    -> FROM Employee
    -> GROUP BY branch_id
    -> HAVING SUM(salary) > 250000
    -> ORDER BY SUM(salary) ASC
    -> LIMIT 1;
+-------------+-----------+
| SUM(salary) | branch_id |
+-------------+-----------+
|      262000 |         2 |
+-------------+-----------+
1 row in set (0.00 sec)
```

LIMIT 5 =) Show top 5 records only

Reason =) Screen capacity is small the data contains millions of records.

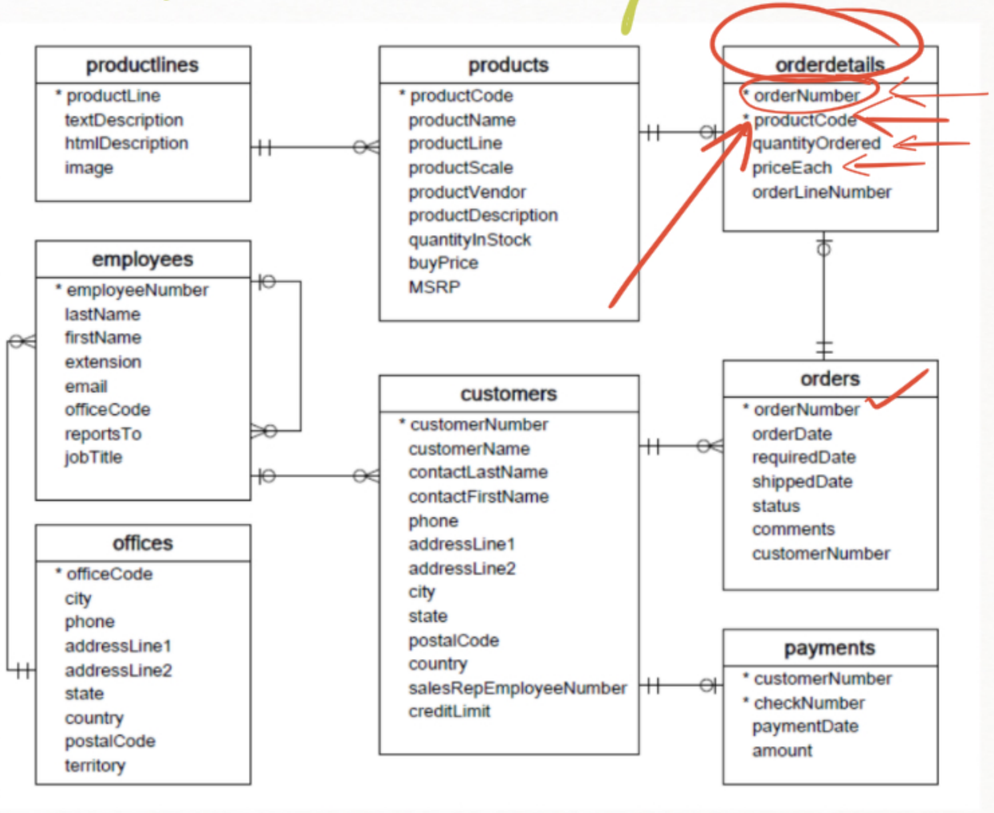
★ Q1. All the orderNumber and the total amount spent when total amount spent > 1000

	Q	Price
Coca cola	5	25
Red Bull	10	200

10X200 = 2000

+ 5X25

Min

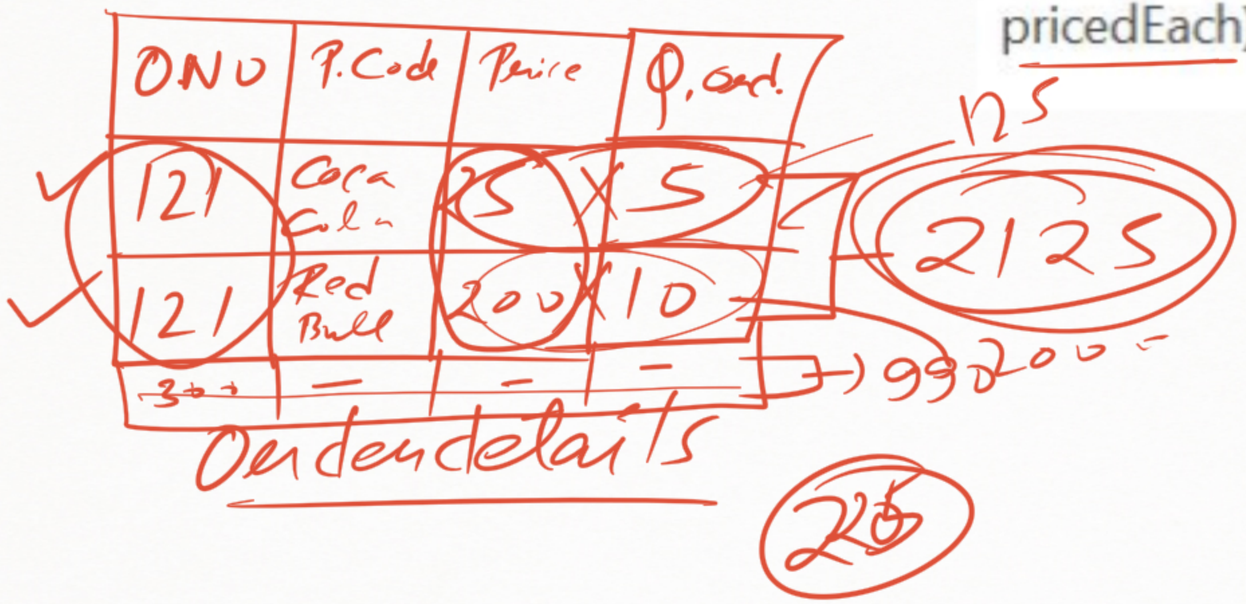


From syed khundmir to Me: (Privately)

```
select orderNumber,(priceEach *
quantity_demanded) AS total_amount from
orderDetails where total_amount > 1000;
```

From Ankit Pal to Me: (Privately)

```
select orderNumber,(quantityordered *
pricedEach) From orderdetails group by
ordernumber having (quantityordered *
pricedEach) >1000
```



From Shouaib Mujawar to Me: (Privately)

```
SELECT ordernumber sum(priceeach) AS
totalamount FROM orderdetails GROUP BY
ordernumber HAVING totalamount>1000;
```

```
mysql> select orderNumber,SUM(quantityordered*priceEach)
    -> FROM orderdetails
    -> GROUP BY orderNumber
    -> HAVING SUM(quantityordered*priceEach) <1000;
+-------------+-----------------------------------+
| orderNumber | SUM(quantityordered*priceEach) |
+-------------+-----------------------------------+
|       10408 |                            615.45 |
+-------------+-----------------------------------+
1 row in set (0.00 sec)
```

emp_id	first_name	last_name	birth_date	sex	salary	super_id	branch_id
100	David	Wallace	1967-11-17	M	250,000	NULL	1
101	Jan	Levinson	1961-05-11	F	110,000	100	1
102	Michael	Scott	1964-03-15	M	75,000	100	2
103	Angela	Martin	1971-06-25	F	63,000	102	2
104	Kelly	Kapoor	1980-02-05	F	55,000	102	2
105	Stanley	Hudson	1958-02-19	M	69,000	102	2
106	Josh	Porter	1969-09-05	M	78,000	100	3
107	Andy	Bernard	1973-07-22	M	65,000	106	3
108	Jim	Halpert	1978-10-01	M	71,000	106	3

emp

Q1. # Print f_name of employees who earns more than everybody in branch=2

Nested Queries

Non-Correlated

Select f_name from employee where salary > (Select Max(salary) from employee where branch=2)

In Non Correlated Query, the Nested Query runs first and then the Outer query runs.

```
Select f_name from emp
where sal > ( Select Max(salary) from
              emp where be_id
                        = 2.
            )
```

Outer Query

Inner Query