Lecture 3

SQL Queries

SQL Queries

- Extract information from the database
- Example: Faculty database
    - Which is the name of the student with the id = 256?
    - Which is the name of the professor that has the highest number of students on his / her course?
    - How many students are enrolled in the Databases course in each specializations?
    - …
- Such questions refers to the data that is kept in a DMBS are called ***queries***.
- ***→ Query language***

SQL Queries

SQL – Structured Query Language

- SELECT Name, Surname, Age FROM Student WHERE Age>19

Relational Algebra

- $\boldsymbol{\pi}_{\text{Name, Surname, Age}}(\boldsymbol{\sigma}_{\text{Age>19}}(\text{Student}))$

- SQL allows to query a database
- SQL is the standard language for relational DBMS (due to ANSI standard)
- SQL was initially developed at IBM by Donald D. Chamberlin, Raymond F. Boyce (using the relational model from Edgar F. Codd) - 1970s - SEQUEL (Structured English Query Language)
- SQL standard revisions: SQL-86, SQL-89, SQL-92, SQL-1999, SQL-2003, SQL-2006, SQL-2008, SQL-2011, SQL-2016, SQL-2019 (multidimensional arrays)

SQL Queries

```
SELECT select_list [ INTO new_table]
FROM from_list
WHERE qualification
GROUP BY group_by_list
HAVING group_qualification
ORDER BY order_by_expression [ASC | DESC]
```

A query is a request for data / information from a database table or combination of tables.

Basic SELECT query

SELECT [DISTINCT] select_list

FROM from_list

WHERE qualification

select_list – list of attributes (expressions) from relations / tables in the *from_list*
from_list – list of relation / table names; can be followed by a range variable
qualification - conditions on the data from the relations / tables in the **from-list**
e.g. conditions: *exprression1 operator expr*,ession2, where *operator* ∈ {<,≤,=,>,≥,≠}, and *expression1, expression2* can include attributes, constants, …; logical operators AND, OR, NOT

- The SELECT, FROM clauses - mandatory
- The WHERE clause - optional
- The result-set returned is a table

Conceptual evaluation strategy:

- compute the cross product of tables in the **from-list**
- only the rows from the **select_list** are displayed
- by default, the duplicates are not eliminated (use DISTINCT to avoid the duplicates)
- only the rows that meet the **qualification** are displayed

SQL Queries

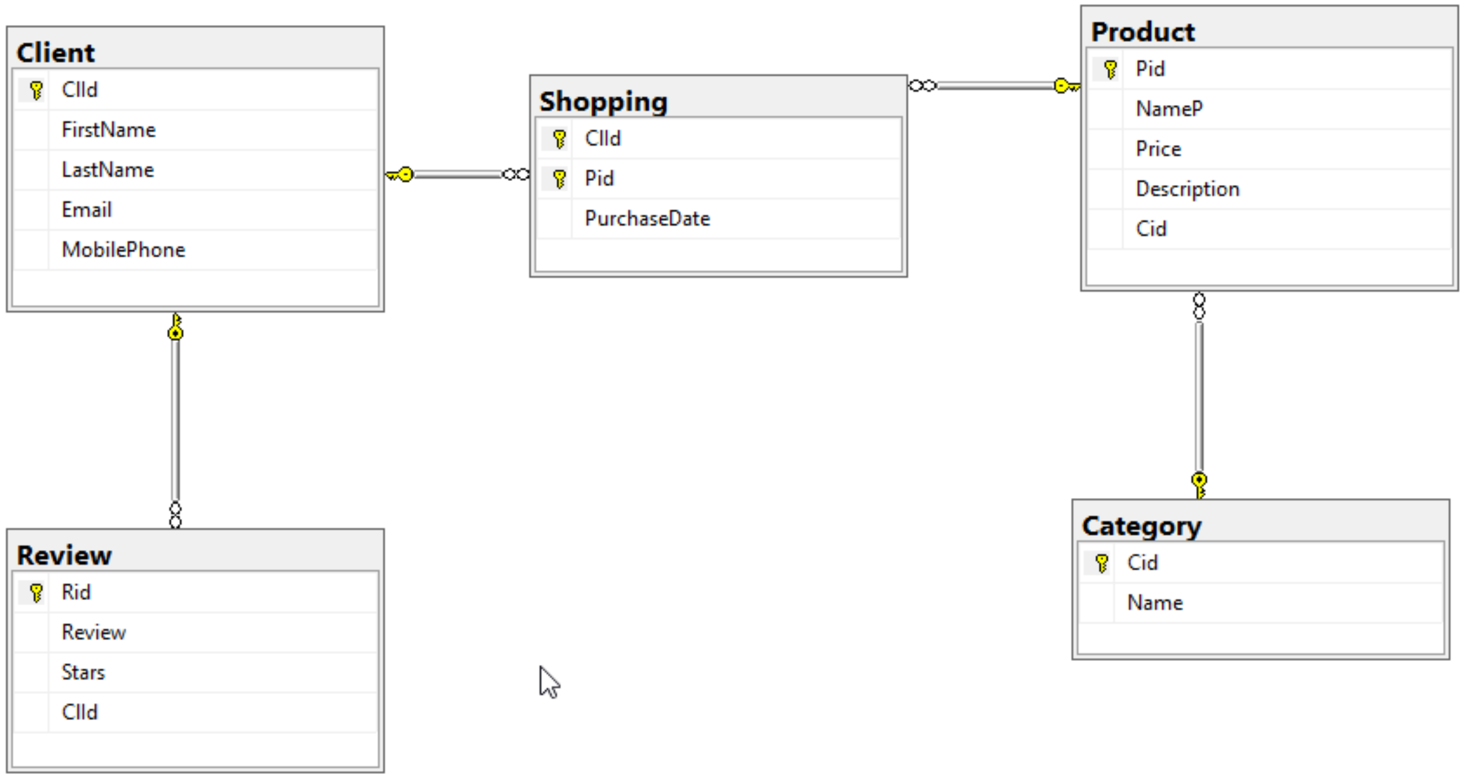
Consider the following code to create the PhotoShop database:

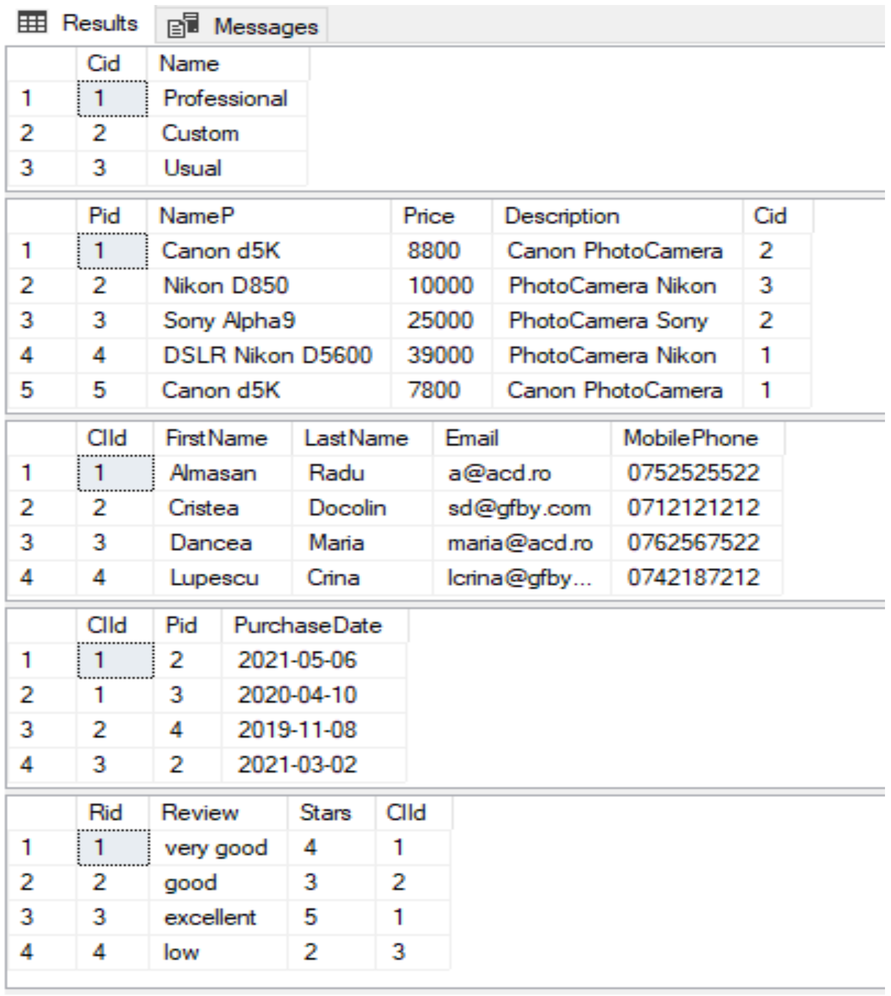
```
-- drop database PhotoShop
CREATE DATABASE PhotoShop
GO
USE PhotoShop
GO

CREATE TABLE Category(
Cid INT PRIMARY KEY IDENTITY,
Name VARCHAR(30))

CREATE TABLE Product(
Pid INT PRIMARY KEY IDENTITY,
NameP VARCHAR(100),
Price INT CHECK(Price>0) NOT NULL,
Description VARCHAR(1000),
Cid INT FOREIGN KEY REFERENCES Category(Cid))

CREATE TABLE Client(
ClId INT PRIMARY KEY IDENTITY,
FirstName VARCHAR(50),
LastName VARCHAR(50),
Email VARCHAR(50),
MobilePhone VARCHAR(50)
)
```

```
CREATE TABLE Shopping(
ClId INT FOREIGN KEY REFERENCES
Client(ClId),
Pid INT FOREIGN KEY REFERENCES
Product(Pid),
PurchaseDate date
CONSTRAINT Pk_Shoppin PRIMARY KEY(ClId,Pid)
)

CREATE TABLE Review(
Rid int primary key identity,
Review varchar(500),
Stars INT CHECK(Stars>0 AND Stars<=5) NOT
NULL,
ClId INT FOREIGN KEY REFERENCES
Client(ClId)
)
```

SQL Queries

Consider the relational schema of the PhotoShop database :





	Cid	Name
1	1	Professional
2	2	Custom
3	3	Usual

	Pid	NameP	Price	Description	Cid
1	1	Canon d5K	8800	Canon PhotoCamera	2
2	2	Nikon D850	10000	PhotoCamera Nikon	3
3	3	Sony Alpha9	25000	PhotoCamera Sony	2
4	4	DSLR Nikon D5600	39000	PhotoCamera Nikon	1
5	5	Canon d5K	7800	Canon PhotoCamera	1

	CIId	FirstName	LastName	Email	MobilePhone
1	1	Almasan	Radu	a@acd.ro	0752525522
2	2	Cristea	Docolin	sd@gfby.com	0712121212
3	3	Dancea	Maria	maria@acd.ro	0762567522
4	4	Lupescu	Crina	lcrina@gfby...	0742187212

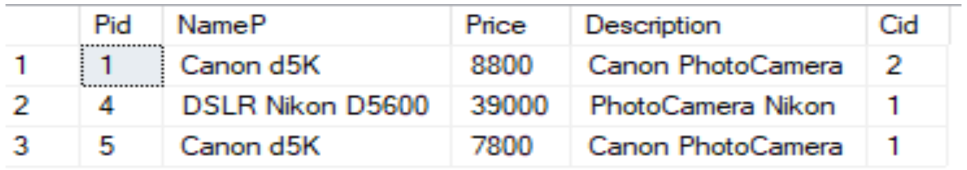
	CIId	Pid	PurchaseDate
1	1	2	2021-05-06
2	1	3	2020-04-10
3	2	4	2019-11-08
4	3	2	2021-03-02

	Rid	Review	Stars	CIId
1	1	very good	4	1
2	2	good	3	2
3	3	excellent	5	1
4	4	low	2	3

SQL Queries- Examples

Find the products with a given name or with the product id greater than a specified value.

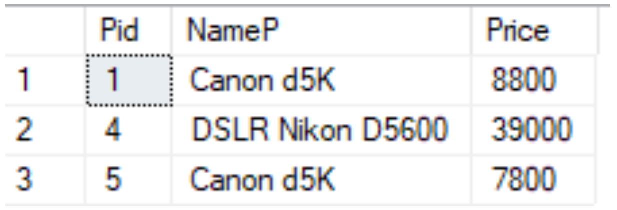
```sql
SELECT *
FROM Product
WHERE NameP = 'Canon d5K' OR Pid>=4
```



	Pid	NameP	Price	Description	Cid
1	1	Canon d5K	8800	Canon PhotoCamera	2
2	4	DSLR Nikon D5600	39000	PhotoCamera Nikon	1
3	5	Canon d5K	7800	Canon PhotoCamera	1

* - all the fields from the table

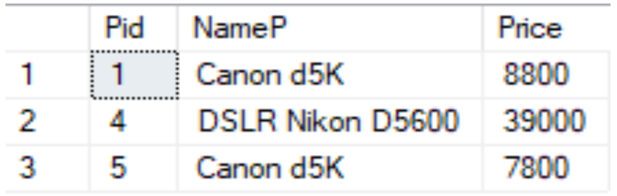
```sql
SELECT Pid, NameP, Price
FROM Product
WHERE NameP = 'Canon d5K' OR Pid>=4
```



	Pid	NameP	Price
1	1	Canon d5K	8800
2	4	DSLR Nikon D5600	39000
3	5	Canon d5K	7800

ALIAS – for table names, columns, expressions

```sql
SELECT P.Pid, P.NameP, P.Price
FROM Product P
WHERE NameP = 'Canon d5K' OR Pid>=4
```

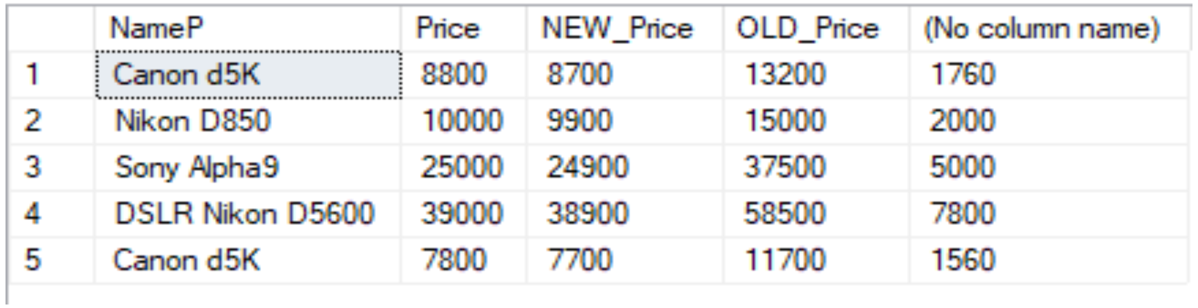


	Pid	NameP	Price
1	1	Canon d5K	8800
2	4	DSLR Nikon D5600	39000
3	5	Canon d5K	7800

SQL Queries – Expressions and Strings

- **AS, = -** are used to rename the fields from result (also arithmetical expressions)

```
SELECT   NameP,   Price,   Price-100   as   NEW_Price,
OLD_Price=Price*2- Price/2, Price/5
FROM Product
```



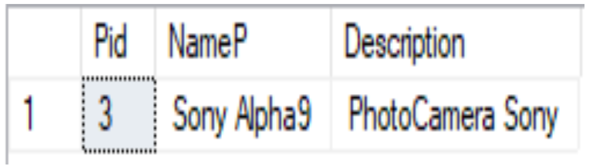
	NameP	Price	NEW_Price	OLD_Price	(No column name)
1	Canon d5K	8800	8700	13200	1760
2	Nikon D850	10000	9900	15000	2000
3	Sony Alpha9	25000	24900	37500	5000
4	DSLR Nikon D5600	39000	38900	58500	7800
5	Canon d5K	7800	7700	11700	1560

- **LIKE** - is used for comparations on the strings / text
  - **_** - represent any character, but only one
  - **%** - stands for 0 or more arbitrary characters

SQL Queries – Strings

Find the products that contains *Alpha* in their names and the keyword *Photo* in the beginning of their description.

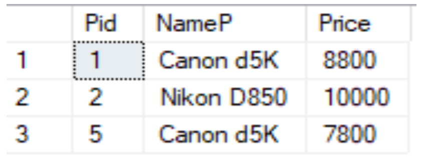
```sql
SELECT Pid, NameP, Description
FROM Product
WHERE NameP LIKE '%Alpha%' AND Description LIKE 'Photo%'
```



	Pid	NameP	Description
1	3	Sony Alpha9	PhotoCamera Sony

Find the products that have the name starting with *C* and having at least 2 characters, or the ones with the price between 500 and 10000.

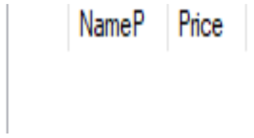
```sql
SELECT Pid, NameP, Price
FROM Product
WHERE NameP = 'C_%' OR Price BETWEEN 500 AND 10000
```



	Pid	NameP	Price
1	1	Canon d5K	8800
2	2	Nikon D850	10000
3	5	Canon d5K	7800

Find the products that have the Canon or Nikon or Sony.

```sql
SELECT P.NameP, P.Price
FROM Product P
WHERE P.NameP IN ('Canon', 'Nikon', 'Sony')
```

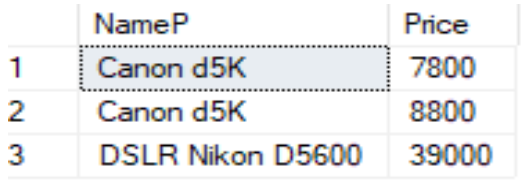


NameP	Price

SQL Queries – DISTINCT, TOP

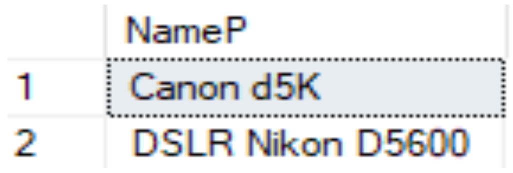
- **DISTINCT** – eliminates the duplicates

```
SELECT DISTINCT NameP, Price
FROM Product
WHERE NameP = 'Canon d5K' OR Pid>=4
```



	NameP	Price
1	Canon d5K	7800
2	Canon d5K	8800
3	DSLR Nikon D5600	39000

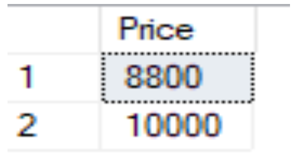
```
SELECT DISTINCT NameP
FROM Product
WHERE NameP = 'Canon d5K' OR Pid>=4
```



	NameP
1	Canon d5K
2	DSLR Nikon D5600

- **TOP n** – displays the first *n* rows from the result set (or less, if *n* greater than the total number of rows)

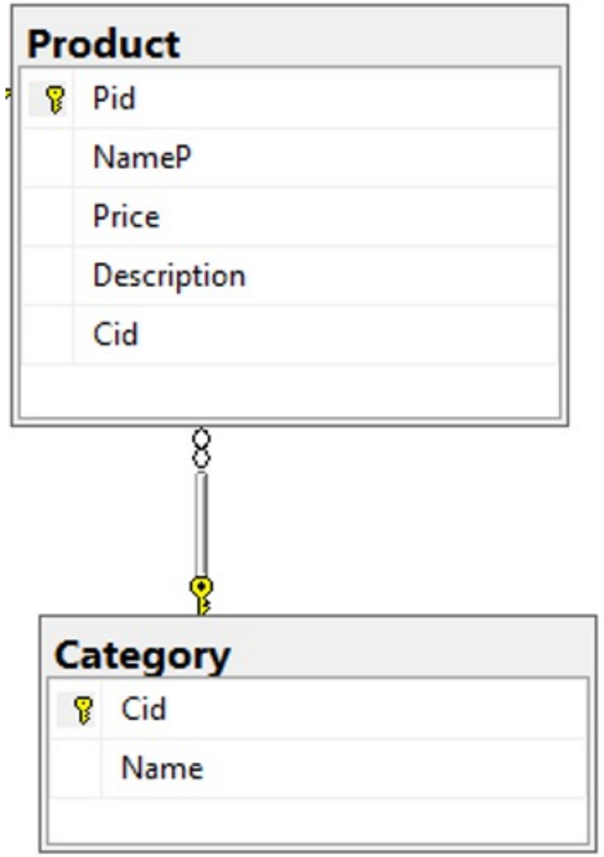
```
SELECT Top 2 Price
FROM Product
```



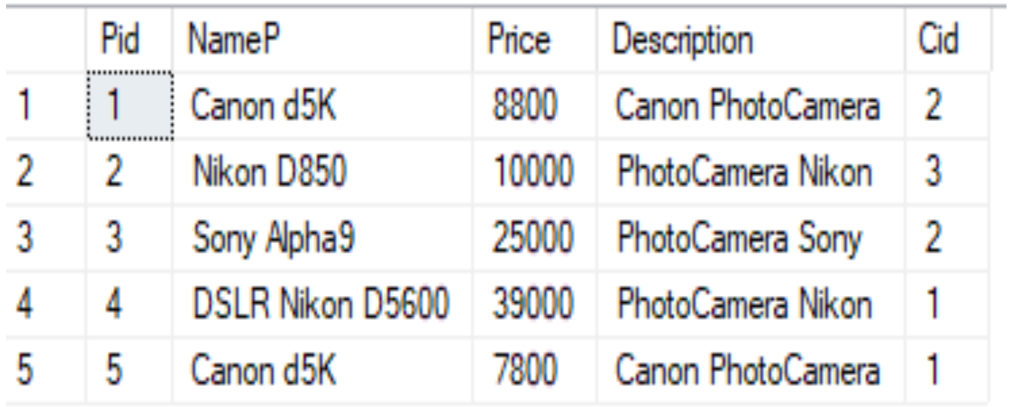
	Price
1	8800
2	10000

SQL Queries – JOIN operations

Consider the relation 1:n – Category – Product, with the tuples:

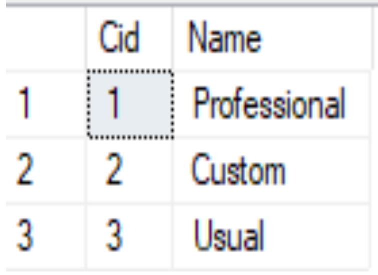


Product



	Pid	NameP	Price	Description	Cid
1	1	Canon d5K	8800	Canon PhotoCamera	2
2	2	Nikon D850	10000	PhotoCamera Nikon	3
3	3	Sony Alpha9	25000	PhotoCamera Sony	2
4	4	DSLR Nikon D5600	39000	PhotoCamera Nikon	1
5	5	Canon d5K	7800	Canon PhotoCamera	1

Category

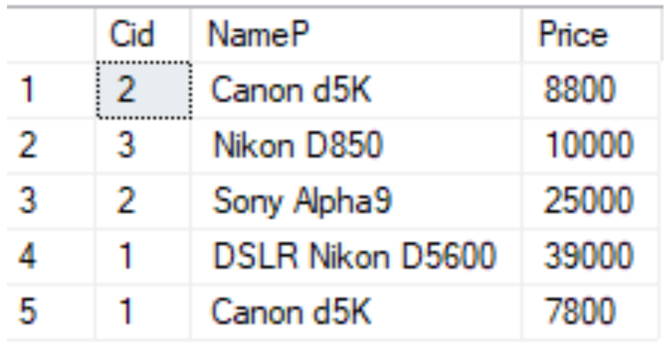


	Cid	Name
1	1	Professional
2	2	Custom
3	3	Usual

SQL Queries – JOIN operations

- Find the products from each category.

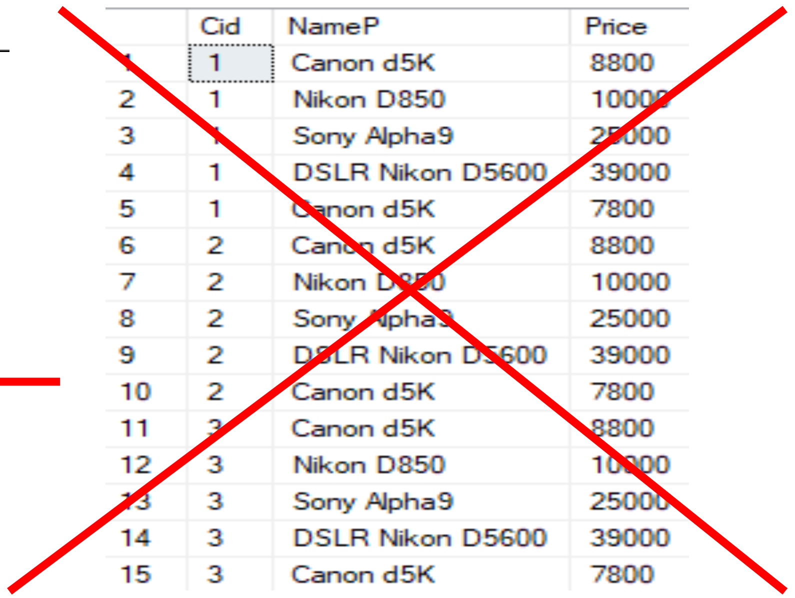
```
-- with WHERE and equivalent with INNER JOIN
SELECT c.Cid, p.NameP, p.Price
FROM Category c, Product p
WHERE c.Cid=p.Cid

-- INNER JOIN
SELECT c.Cid, p.NameP, p.Price
FROM Category c INNER JOIN Product p ON c.Cid=p.Cid
```



	Cid	NameP	Price
1	2	Canon d5K	8800
2	3	Nikon D850	10000
3	2	Sony Alpha9	25000
4	1	DSLR Nikon D5600	39000
5	1	Canon d5K	7800

- Find the products from each category. – **NOT GOOD**

```
SELECT c.Cid, p.NameP, p.Price
FROM Category c, Product p
```

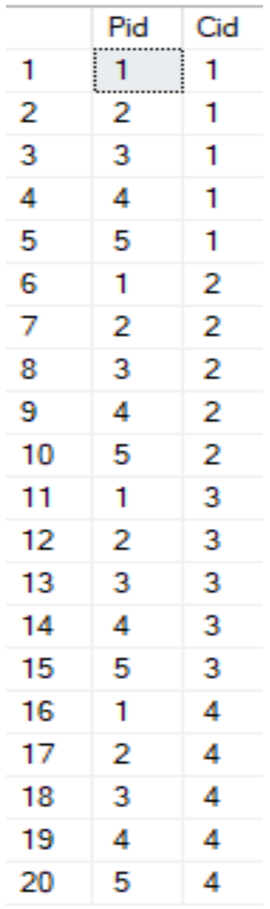


	Cid	NameP	Price
1	1	Canon d5K	8800
2	1	Nikon D850	10000
3	1	Sony Alpha9	25000
4	1	DSLR Nikon D5600	39000
5	1	Canon d5K	7800
6	2	Canon d5K	8800
7	2	Nikon D850	10000
8	2	Sony Alpha9	25000
9	2	DSLR Nikon D5600	39000
10	2	Canon d5K	7800
11	3	Canon d5K	8800
12	3	Nikon D850	10000
13	3	Sony Alpha9	25000
14	3	DSLR Nikon D5600	39000
15	3	Canon d5K	7800

SQL Queries – JOIN operations

- **CROSS JOIN** – the carthesian product (all the combinations of the fields)

```
select P.Pid, C.Cid
from Product P CROSS JOIN Category C
```



	Pid	Cid
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	1	2
7	2	2
8	3	2
9	4	2
10	5	2
11	1	3
12	2	3
13	3	3
14	4	3
15	5	3
16	1	4
17	2	4
18	3	4
19	4	4
20	5	4

SQL Queries – example

Example on the schema:

- Client [<u>Clid</u>, FirstName, LastName, Email, MobilePhone]
- Product [<u>Pid</u>, NameP, Price, Description, Cid]
- Shopping [<u>Clid</u>, <u>Pid,</u> PurchaseDate]

- Display the first name and the last name of the clients that have bought the product with the Pid=2.

```
SELECT C.Firstname, C.Lastname
FROM Client C, Shopping S
WHERE C.Clid=S.Clid AND S.Pid=2
```

Client

Clid	FirstName	LastName	Email	MobilePhone
1	Almasan	Radu	a@acd.ro	0752525522
2	Cristea	Docolin	sd@gfby.com	0712121212
3	Dancea	Maria	maria@acd.ro	0762567522
4	Lupescu	Crina	lcrina@gfby.com	0742187212

Shopping

Clid	Pid	PurchaseDate
1	2	2021-05-06
1	3	2020-04-10
2	4	2019-11-08
3	2	2021-03-02

SQL Queries – example

- Compute the cross product of tables *Client* and *Shopping*

Clid appears in both *Client* and *Shopping* tables, so it will be qualified (due to the WHERE clause)

Clid	FirstName	LastName	Email	MobilePhone	Clid	Pid	PurchaseDate
1	Almasan	Radu	a@acd.ro	0752525522	1	2	2021-05-06
1	Almasan	Radu	a@acd.ro	0752525522	1	3	2020-04-10
1	Almasan	Radu	a@acd.ro	0752525522	2	4	2019-11-08
1	Almasan	Radu	a@acd.ro	0752525522	3	2	2021-03-02
2	Cristea	Docolin	sd@gfby.com	0712121212	1	2	2021-05-06
2	Cristea	Docolin	sd@gfby.com	0712121212	1	3	2020-04-10
2	Cristea	Docolin	sd@gfby.com	0712121212	2	4	2019-11-08
2	Cristea	Docolin	sd@gfby.com	0712121212	3	2	2021-03-02
3	Dancea	Maria	maria@acd.ro	0762567522	1	2	2021-05-06
3	Dancea	Maria	maria@acd.ro	0762567522	1	3	2020-04-10
3	Dancea	Maria	maria@acd.ro	0762567522	2	4	2019-11-08
3	Dancea	Maria	maria@acd.ro	0762567522	3	2	2021-03-02
4	Lupescu	Crina	lcrina@gfby.com	0742187212	1	2	2021-05-06
4	Lupescu	Crina	lcrina@gfby.com	0742187212	1	3	2020-04-10
4	Lupescu	Crina	lcrina@gfby.com	0742187212	2	4	2019-11-08
4	Lupescu	Crina	lcrina@gfby.com	0742187212	3	2	2021-03-02

SQL Queries – example

- Remove the rows in the cross product that don't satisfy the condition C.Clid=S.Clid AND S.Pid=2

C.Clid=S.Clid

Clid	FirstName	LastName	Email	MobilePhone	Clid	Pid	PurchaseDate
1	Almasan	Radu	a@acd.ro	0752525522	1	2	2021-05-06
1	Almasan	Radu	a@acd.ro	0752525522	1	3	2020-04-10
1	Almasan	Radu	a@acd.ro	0752525522	2	4	2019-11-08
1	Almasan	Radu	a@acd.ro	0752525522	3	2	2021-03-02
2	Cristea	Docolin	sd@gfby.com	0712121212	1	2	2021-05-06
2	Cristea	Docolin	sd@gfby.com	0712121212	1	3	2020-04-10
2	Cristea	Docolin	sd@gfby.com	0712121212	2	4	2019-11-08
2	Cristea	Docolin	sd@gfby.com	0712121212	3	2	2021-03-02
3	Dancea	Maria	maria@acd.ro	0762567522	1	2	2021-05-06
3	Dancea	Maria	maria@acd.ro	0762567522	1	3	2020-04-10
3	Dancea	Maria	maria@acd.ro	0762567522	2	4	2019-11-08
3	Dancea	Maria	maria@acd.ro	0762567522	3	2	2021-03-02
4	Lupescu	Crina	lcrina@gfby.com	0742187212	1	2	2021-05-06
4	Lupescu	Crina	lcrina@gfby.com	0742187212	1	3	2020-04-10
4	Lupescu	Crina	lcrina@gfby.com	0742187212	2	4	2019-11-08
4	Lupescu	Crina	lcrina@gfby.com	0742187212	3	2	2021-03-02

SQL Queries – example

- Remove the rows in the cross product that don't satisfy the condition C.Clid=S.Clid AND S.Pid=2

S.Pid=2

Clid	FirstName	LastName	Email	MobilePhone	Clid	Pid	PurchaseDate
1	Almasan	Radu	a@acd.ro	0752525522	1	2	2021-05-06
1	Almasan	Radu	a@acd.ro	0752525522	1	3	2020-04-10
1	Almasan	Radu	a@acd.ro	0752525522	2	4	2019-11-08
1	Almasan	Radu	a@acd.ro	0752525522	3	2	2021-03-02
2	Cristea	Docolin	sd@gfby.com	0712121212	1	2	2021-05-06
2	Cristea	Docolin	sd@gfby.com	0712121212	1	3	2020-04-10
2	Cristea	Docolin	sd@gfby.com	0712121212	2	4	2019-11-08
2	Cristea	Docolin	sd@gfby.com	0712121212	3	2	2021-03-02
3	Dancea	Maria	maria@acd.ro	0762567522	1	2	2021-05-06
3	Dancea	Maria	maria@acd.ro	0762567522	1	3	2020-04-10
3	Dancea	Maria	maria@acd.ro	0762567522	2	4	2019-11-08
3	Dancea	Maria	maria@acd.ro	0762567522	3	2	2021-03-02
4	Lupescu	Crina	lcrina@gfby.com	0742187212	1	2	2021-05-06
4	Lupescu	Crina	lcrina@gfby.com	0742187212	1	3	2020-04-10
4	Lupescu	Crina	lcrina@gfby.com	0742187212	2	4	2019-11-08
4	Lupescu	Crina	lcrina@gfby.com	0742187212	3	2	2021-03-02

SQL Queries – example

- Remove the rows in the cross product that don't satisfy the condition C.Clid=S.Clid AND S.Pid=2

C.Clid=S.Clid AND S.Pid=2

Clid	FirstName	LastName	Email	MobilePhone	Clid	Pid	PurchaseDate
1	Almasan	Radu	a@acd.ro	0752525522	1	2	2021-05-06
1	Almasan	Radu	a@acd.ro	0752525522	1	3	2020-04-10
1	Almasan	Radu	a@acd.ro	0752525522	2	4	2019-11-08
1	Almasan	Radu	a@acd.ro	0752525522	3	2	2021-03-02
2	Cristea	Docolin	sd@gfby.com	0712121212	1	2	2021-05-06
2	Cristea	Docolin	sd@gfby.com	0712121212	1	3	2020-04-10
2	Cristea	Docolin	sd@gfby.com	0712121212	2	4	2019-11-08
2	Cristea	Docolin	sd@gfby.com	0712121212	3	2	2021-03-02
3	Dancea	Maria	maria@acd.ro	0762567522	1	2	2021-05-06
3	Dancea	Maria	maria@acd.ro	0762567522	1	3	2020-04-10
3	Dancea	Maria	maria@acd.ro	0762567522	2	4	2019-11-08
3	Dancea	Maria	maria@acd.ro	0762567522	3	2	2021-03-02
4	Lupescu	Crina	lcrina@gfby.com	0742187212	1	2	2021-05-06
4	Lupescu	Crina	lcrina@gfby.com	0742187212	1	3	2020-04-10
4	Lupescu	Crina	lcrina@gfby.com	0742187212	2	4	2019-11-08
4	Lupescu	Crina	lcrina@gfby.com	0742187212	3	2	2021-03-02

SQL Queries – example

- Remove the rows in the cross product that don't satisfy the condition C.Clid=S.Clid AND S.Pid=2

  C.Clid=S.Clid AND S.Pid=2 - Only the common ones remain

Clid	FirstName	LastName	Email	MobilePhone	Clid	Pid	PurchaseDate
1	Almasan	Radu	a@acd.ro	0752525522	1	2	2021-05-06
3	Dancea	Maria	maria@acd.ro	0762567522	3	2	2021-03-02

- Remove the columns that don't appear in the final result of the query

FirstName	LastName
Almasan	Radu
Dancea	Maria

```
SELECT C.Firstname, C.Lastname
FROM Client C, Shopping S
WHERE C.Clid=S.Clid AND S.Pid=2
```

Display the first name and the last name of the clients that have bought the product with the Pid=2

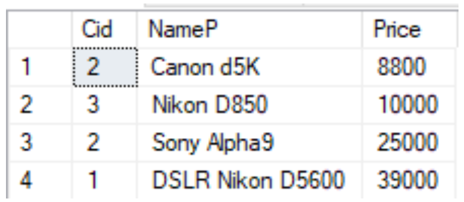
SQL Queries – JOIN operations

EXAMPLE: Find the products from each category that have the price greater than a given value.

- **INNER JOIN** – only the records that “communicate / appear” in both of the tables (no NULL values in the result set)

inner join: table1 [alias] **[INNER] JOIN** table2 [alias] **ON** condition

```sql
SELECT c.Cid, p.NameP, p.Price
FROM Category c INNER JOIN Product p ON c.Cid=p.Cid
WHERE Price>8000
```

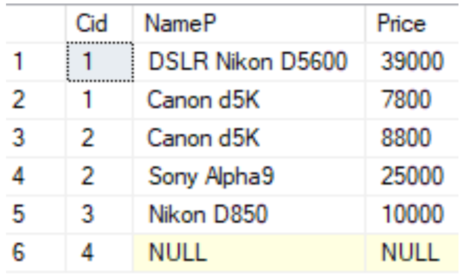


	Cid	NameP	Price
1	2	Canon d5K	8800
2	3	Nikon D850	10000
3	2	Sony Alpha9	25000
4	1	DSLR Nikon D5600	39000

- **LEFT [OUTER] JOIN** – all the records from the left side table even if are “communicating / appearing” or not, in the right side table (possible NULL values for the fields of the right side table in the result set)

left outer join: table1 [alias] **LEFT [OUTER] JOIN** table2 [alias] **ON** condition

```sql
SELECT c.Cid, p.NameP, p.Price
FROM Category c LEFT OUTER JOIN Product p ON c.Cid=p.Cid
```



	Cid	NameP	Price
1	1	DSLR Nikon D5600	39000
2	1	Canon d5K	7800
3	2	Canon d5K	8800
4	2	Sony Alpha9	25000
5	3	Nikon D850	10000
6	4	NULL	NULL

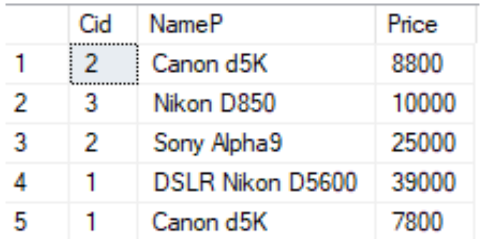
SQL Queries – JOIN operations

EXAMPLE: Find the products from each category that have the price greater than a given value.

- **RIGHT [OUTER] JOIN** – all the records from the right side table even if are “communicating / appearing” or not, in the left side table (possible NULL values for the fields of the left side table in the result set)

right outer join: table1 [alias] **RIGHT [OUTER] JOIN** table2 [alias] **ON** condition

```
SELECT c.Cid, p.NameP, p.Price
FROM Category c RIGHT OUTER JOIN Product p ON c.Cid=p.Cid
```

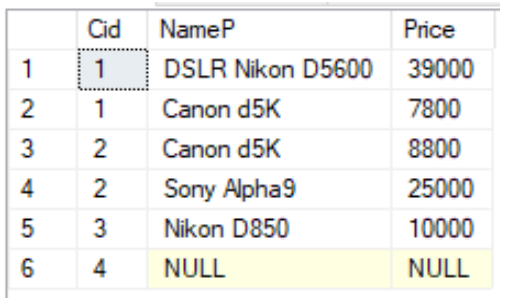


	Cid	NameP	Price
1	2	Canon d5K	8800
2	3	Nikon D850	10000
3	2	Sony Alpha9	25000
4	1	DSLR Nikon D5600	39000
5	1	Canon d5K	7800

- **FULL [OUTER] JOIN** – all the records from the left and right side tables even if are “communicating / appearing” or not (possible NULL values for the fields of the left and right side tables in the result set) – LEFT OUTER JOIN + RIGHT OUTER JOIN

full outer join: table1 [alias] **FULL [OUTER] JOIN** table2 [alias] **ON** condition

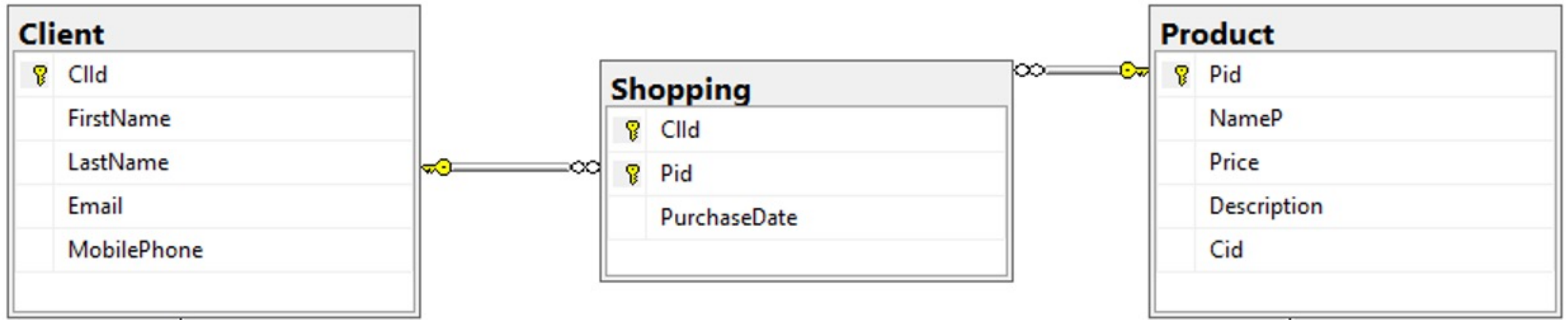
```
SELECT c.Cid, p.NameP, p.Price
FROM Category c FULL OUTER JOIN Product p ON c.Cid=p.Cid
```



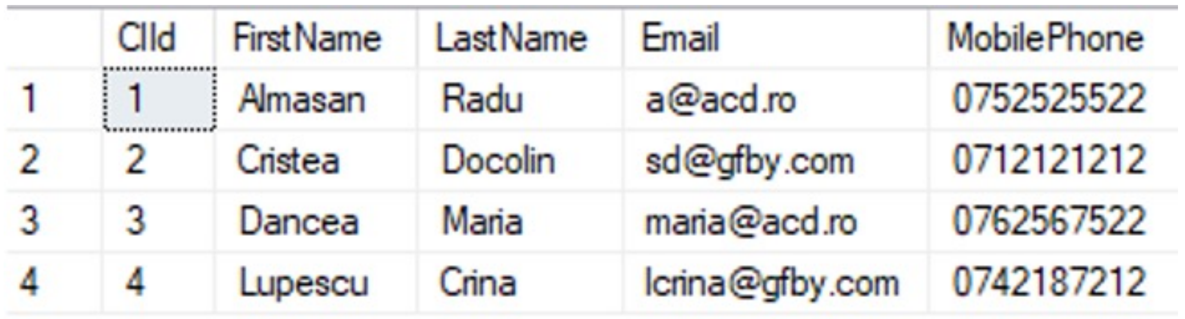
	Cid	NameP	Price
1	1	DSLR Nikon D5600	39000
2	1	Canon d5K	7800
3	2	Canon d5K	8800
4	2	Sony Alpha9	25000
5	3	Nikon D850	10000
6	4	NULL	NULL

SQL Queries – JOIN operations – m-n relationship

Consider the relation m:n – Client – Product, with the tuples:

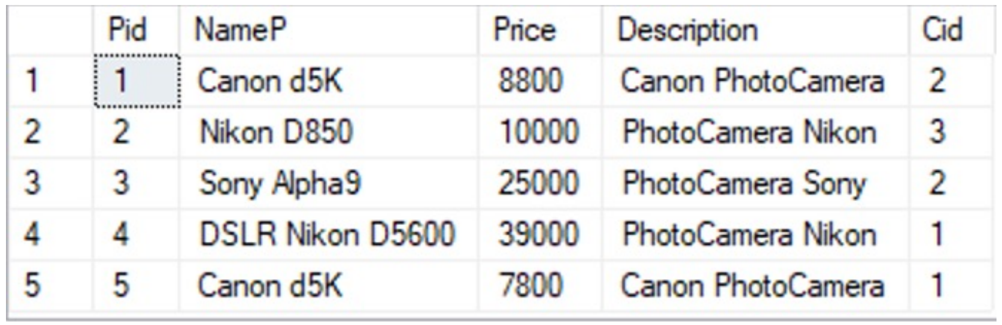


Client



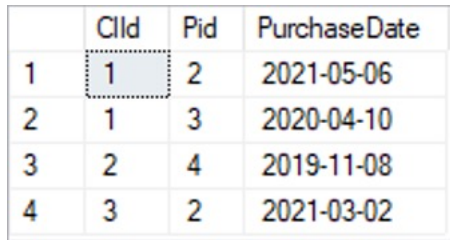
	ClId	FirstName	LastName	Email	MobilePhone
1	1	Almasan	Radu	a@acd.ro	0752525522
2	2	Cristea	Docolin	sd@gfby.com	0712121212
3	3	Dancea	Maria	maria@acd.ro	0762567522
4	4	Lupescu	Crina	lcrina@gfby.com	0742187212

Product



	Pid	NameP	Price	Description	Cid
1	1	Canon d5K	8800	Canon PhotoCamera	2
2	2	Nikon D850	10000	PhotoCamera Nikon	3
3	3	Sony Alpha9	25000	PhotoCamera Sony	2
4	4	DSLR Nikon D5600	39000	PhotoCamera Nikon	1
5	5	Canon d5K	7800	Canon PhotoCamera	1

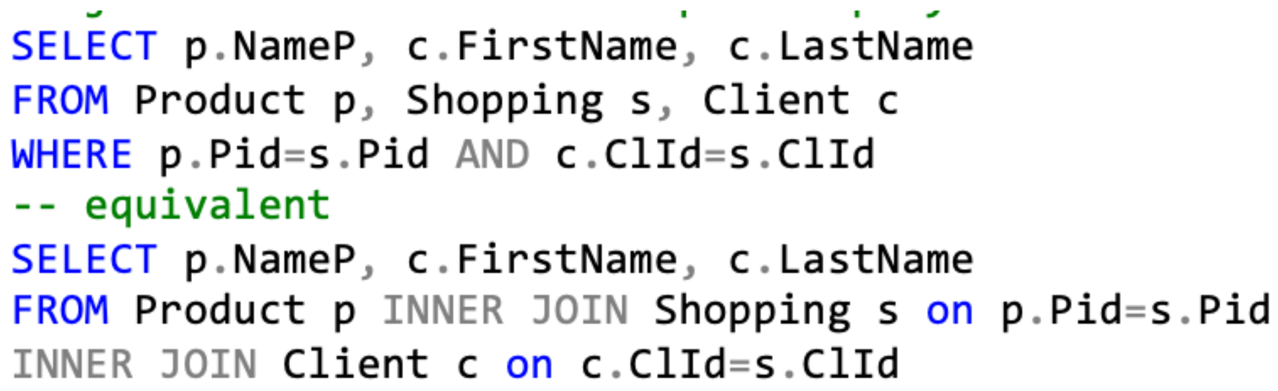
Shopping



	ClId	Pid	PurchaseDate
1	1	2	2021-05-06
2	1	3	2020-04-10
3	2	4	2019-11-08
4	3	2	2021-03-02

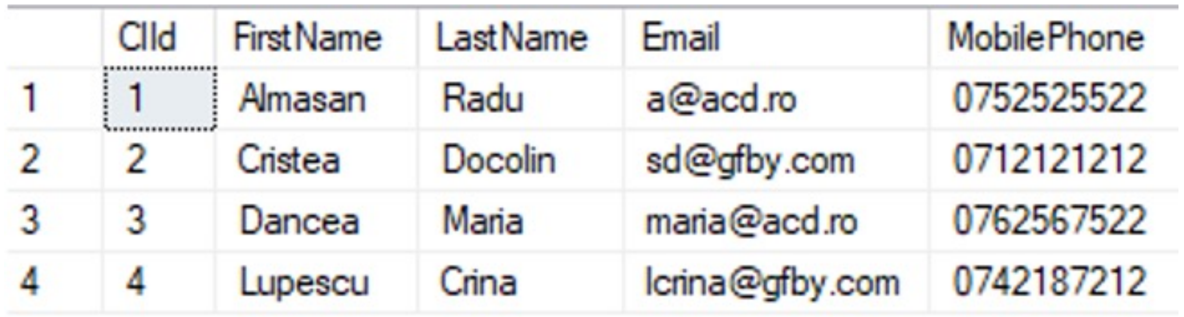
SQL Queries – INNER JOIN

- Display for each client the products bought.



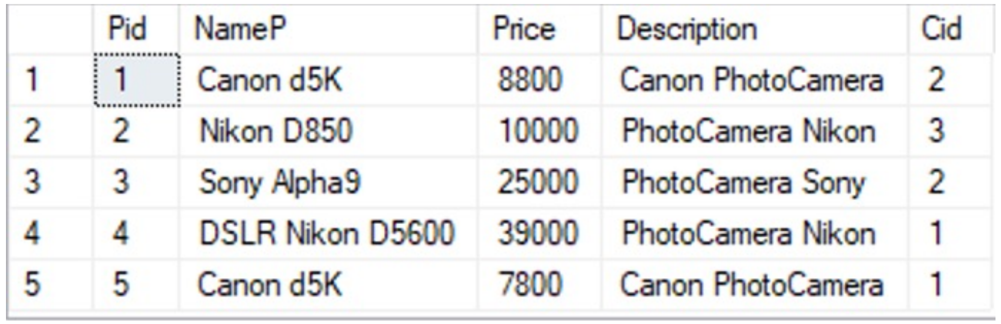
```
SELECT p.NameP, c.FirstName, c.LastName
FROM Product p, Shopping s, Client c
WHERE p.Pid=s.Pid AND c.ClId=s.ClId
-- equivalent
SELECT p.NameP, c.FirstName, c.LastName
FROM Product p INNER JOIN Shopping s on p.Pid=s.Pid
INNER JOIN Client c on c.ClId=s.ClId
```

Client



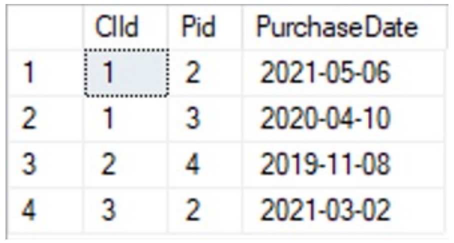
	ClId	FirstName	LastName	Email	MobilePhone
1	1	Almasan	Radu	a@acd.ro	0752525522
2	2	Cristea	Docolin	sd@gfby.com	0712121212
3	3	Dancea	Maria	maria@acd.ro	0762567522
4	4	Lupescu	Crina	lcrina@gfby.com	0742187212

Product



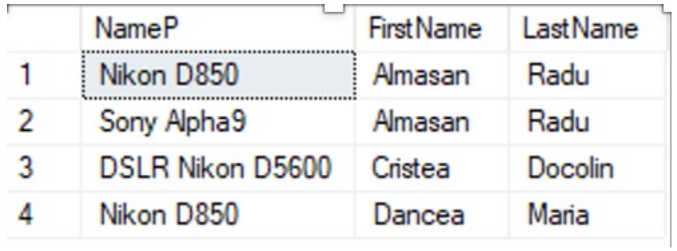
	Pid	NameP	Price	Description	Cid
1	1	Canon d5K	8800	Canon PhotoCamera	2
2	2	Nikon D850	10000	PhotoCamera Nikon	3
3	3	Sony Alpha9	25000	PhotoCamera Sony	2
4	4	DSLR Nikon D5600	39000	PhotoCamera Nikon	1
5	5	Canon d5K	7800	Canon PhotoCamera	1

Shopping



	ClId	Pid	PurchaseDate
1	1	2	2021-05-06
2	1	3	2020-04-10
3	2	4	2019-11-08
4	3	2	2021-03-02

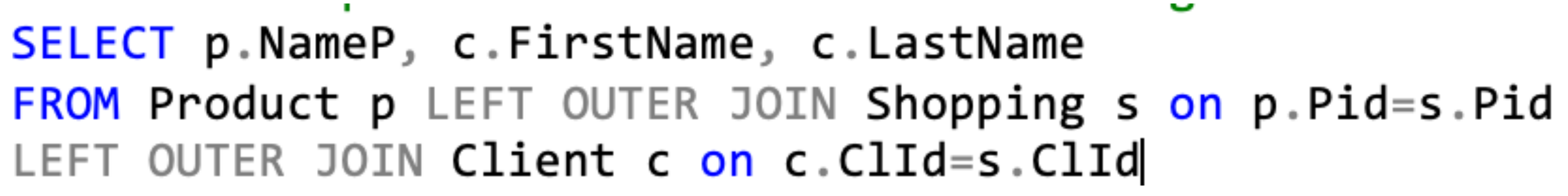
RESULT SET



	NameP	FirstName	LastName
1	Nikon D850	Almasan	Radu
2	Sony Alpha9	Almasan	Radu
3	DSLR Nikon D5600	Cristea	Docolin
4	Nikon D850	Dancea	Maria

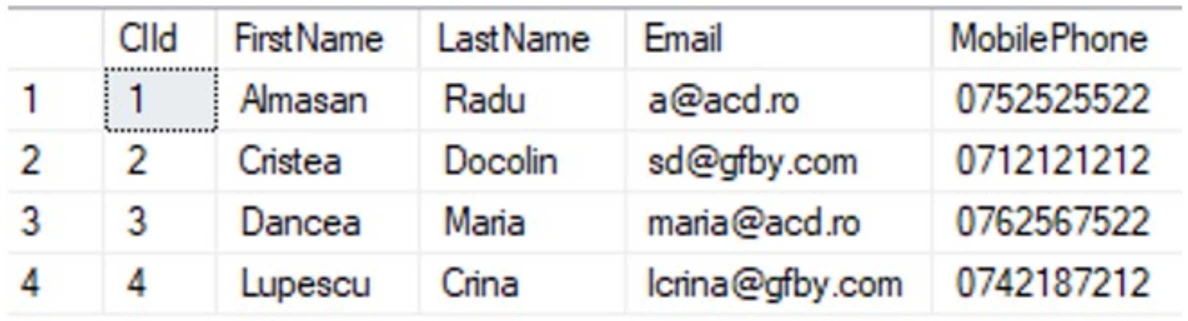
SQL Queries – LEFT OUTER JOIN

- Display for each client the products bought and also the products that haven't been bought yet.



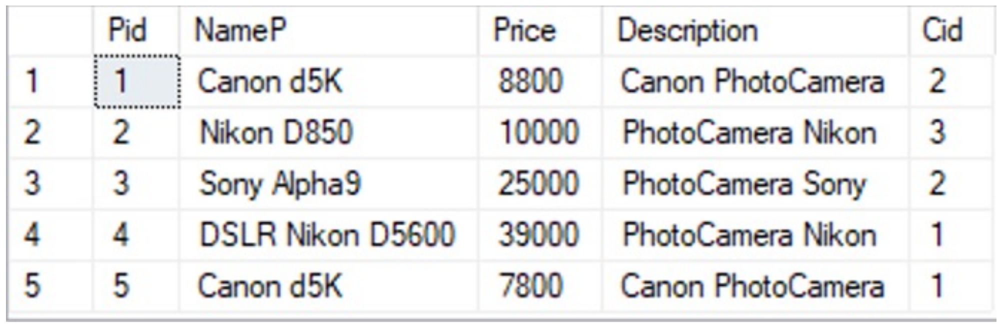
```sql
SELECT p.NameP, c.FirstName, c.LastName
FROM Product p LEFT OUTER JOIN Shopping s on p.Pid=s.Pid
LEFT OUTER JOIN Client c on c.ClId=s.ClId
```

Client



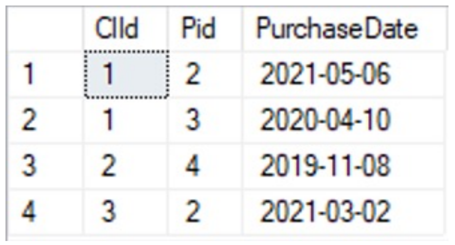
	ClId	FirstName	LastName	Email	MobilePhone
1	1	Almasan	Radu	a@acd.ro	0752525522
2	2	Cristea	Docolin	sd@gfby.com	0712121212
3	3	Dancea	Maria	maria@acd.ro	0762567522
4	4	Lupescu	Crina	lcrina@gfby.com	0742187212

Product



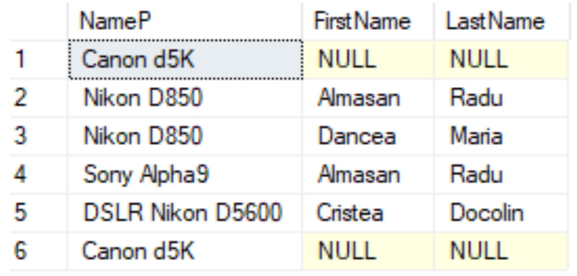
	Pid	NameP	Price	Description	Cid
1	1	Canon d5K	8800	Canon PhotoCamera	2
2	2	Nikon D850	10000	PhotoCamera Nikon	3
3	3	Sony Alpha9	25000	PhotoCamera Sony	2
4	4	DSLR Nikon D5600	39000	PhotoCamera Nikon	1
5	5	Canon d5K	7800	Canon PhotoCamera	1

Shopping



	ClId	Pid	PurchaseDate
1	1	2	2021-05-06
2	1	3	2020-04-10
3	2	4	2019-11-08
4	3	2	2021-03-02

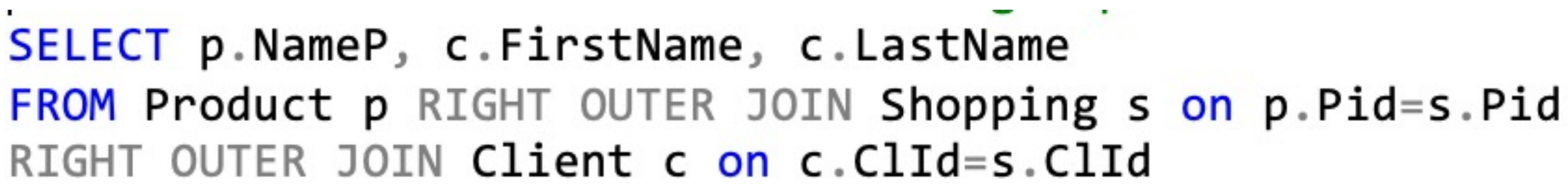
RESULT SET



	NameP	FirstName	LastName
1	Canon d5K	NULL	NULL
2	Nikon D850	Almasan	Radu
3	Nikon D850	Dancea	Maria
4	Sony Alpha9	Almasan	Radu
5	DSLR Nikon D5600	Cristea	Docolin
6	Canon d5K	NULL	NULL

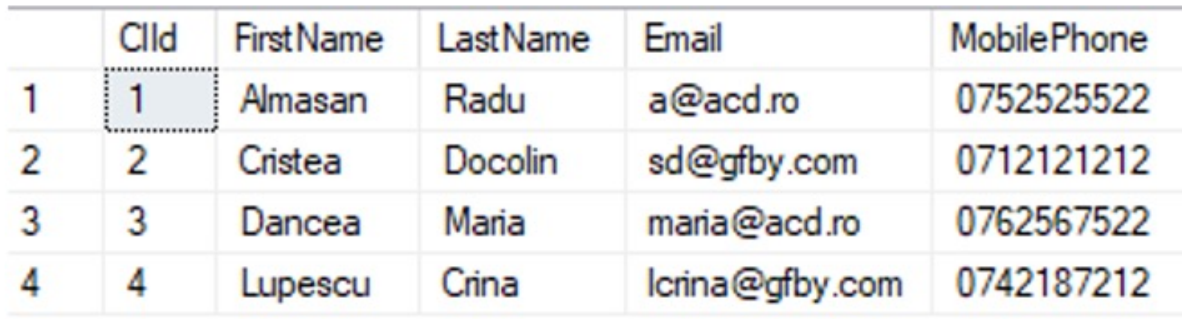
SQL Queries – RIGHT OUTER JOIN

- Display for each client the products bought and also the clients that haven't bought products.



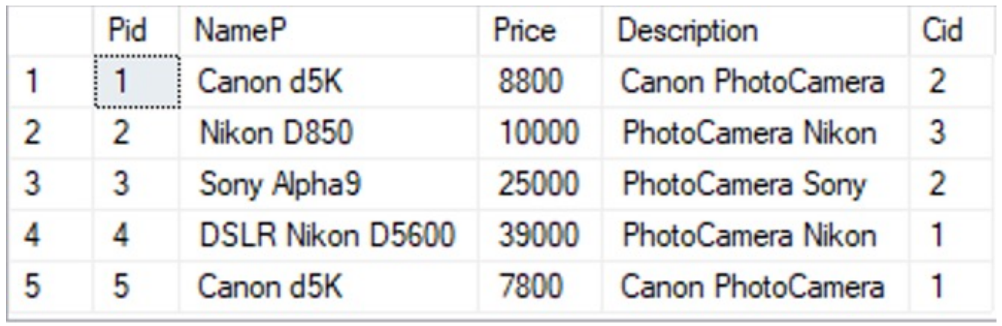
```
SELECT p.NameP, c.FirstName, c.LastName
FROM Product p RIGHT OUTER JOIN Shopping s on p.Pid=s.Pid
RIGHT OUTER JOIN Client c on c.ClId=s.ClId
```

Client



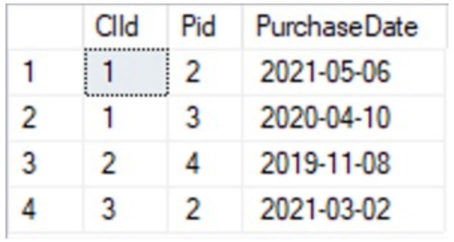
	ClId	FirstName	LastName	Email	MobilePhone
1	1	Almasan	Radu	a@acd.ro	0752525522
2	2	Cristea	Docolin	sd@gfby.com	0712121212
3	3	Dancea	Maria	maria@acd.ro	0762567522
4	4	Lupescu	Crina	lcrina@gfby.com	0742187212

Product



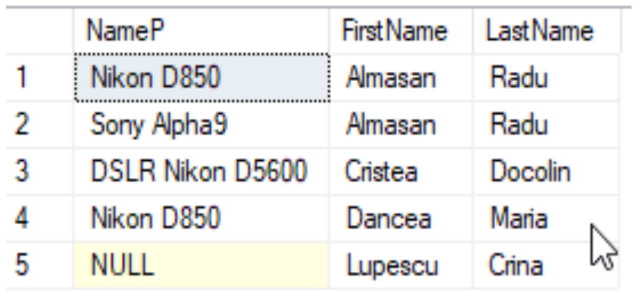
	Pid	NameP	Price	Description	Cid
1	1	Canon d5K	8800	Canon PhotoCamera	2
2	2	Nikon D850	10000	PhotoCamera Nikon	3
3	3	Sony Alpha9	25000	PhotoCamera Sony	2
4	4	DSLR Nikon D5600	39000	PhotoCamera Nikon	1
5	5	Canon d5K	7800	Canon PhotoCamera	1

Shopping



	ClId	Pid	PurchaseDate
1	1	2	2021-05-06
2	1	3	2020-04-10
3	2	4	2019-11-08
4	3	2	2021-03-02

RESULT SET



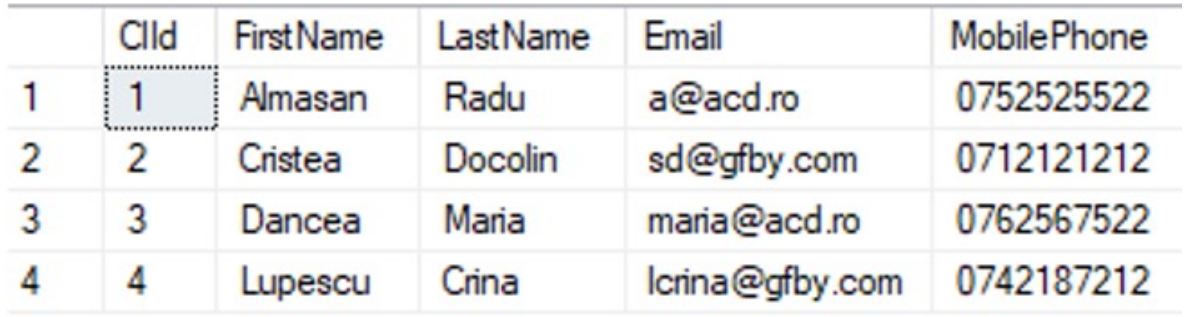
	NameP	FirstName	LastName
1	Nikon D850	Almasan	Radu
2	Sony Alpha9	Almasan	Radu
3	DSLR Nikon D5600	Cristea	Docolin
4	Nikon D850	Dancea	Maria
5	NULL	Lupescu	Crina

SQL Queries – FULL OUTER JOIN

- Display for each client the products bought and also the products that haven't been bought and the clients that haven't bought products (LEFT OUTER JOIN + RIGHT OUTER JOIN).

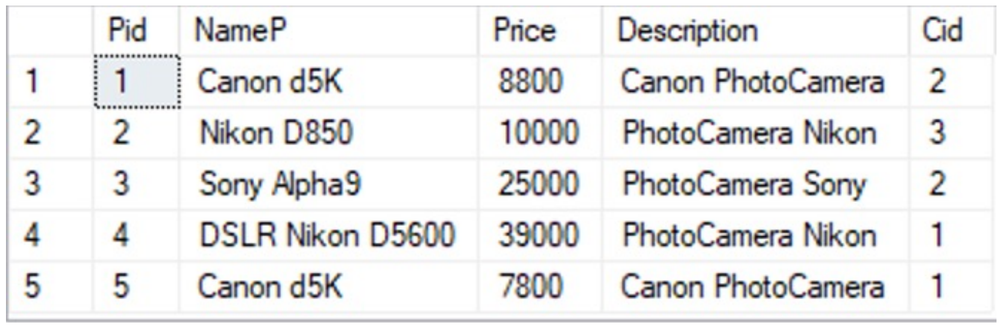
```
SELECT p.NameP, c.FirstName, c.LastName
FROM Product p FULL OUTER JOIN Shopping s on p.Pid=s.Pid
FULL OUTER JOIN Client c on c.ClId=s.ClId
```

Client



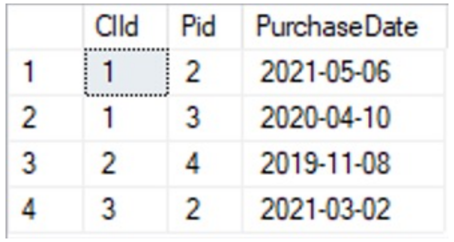
	ClId	FirstName	LastName	Email	MobilePhone
1	1	Almasan	Radu	a@acd.ro	0752525522
2	2	Cristea	Docolin	sd@gfby.com	0712121212
3	3	Dancea	Maria	maria@acd.ro	0762567522
4	4	Lupescu	Crina	lcrina@gfby.com	0742187212

Product



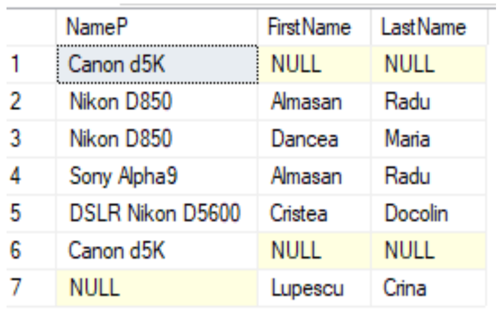
	Pid	NameP	Price	Description	Cid
1	1	Canon d5K	8800	Canon PhotoCamera	2
2	2	Nikon D850	10000	PhotoCamera Nikon	3
3	3	Sony Alpha9	25000	PhotoCamera Sony	2
4	4	DSLR Nikon D5600	39000	PhotoCamera Nikon	1
5	5	Canon d5K	7800	Canon PhotoCamera	1

Shopping



	ClId	Pid	PurchaseDate
1	1	2	2021-05-06
2	1	3	2020-04-10
3	2	4	2019-11-08
4	3	2	2021-03-02

RESULT SET



	NameP	FirstName	LastName
1	Canon d5K	NULL	NULL
2	Nikon D850	Almasan	Radu
3	Nikon D850	Dancea	Maria
4	Sony Alpha9	Almasan	Radu
5	DSLR Nikon D5600	Cristea	Docolin
6	Canon d5K	NULL	NULL
7	NULL	Lupescu	Crina

SQL Queries – Nested Queries

- SELECT … WHERE … (SELECT … (…))
- in the **WHERE** clause

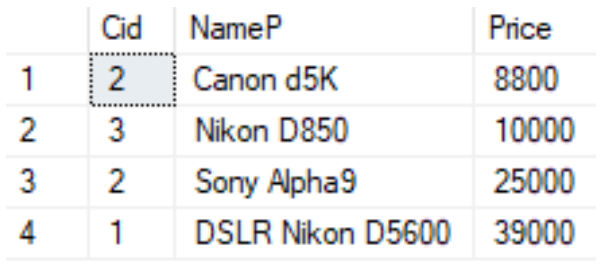
- **IN** operator - tests whether a value belongs to a set of elements; the latter can be explicitly specified or generated by a query.
- **EXISTS** operator - tests whether a set is non-empty.
- **FROM** operator – followed by a table / variable (name)

- **ANY** operator - evaluates to true if the condition is true for at least one item in the subquery result.
- **ALL** operator - evaluates to true if the condition is true for all the items in the subquery result.

SQL Queries – Nested Queries

- Find for each category the products with the price greater than a given value.

```sql
-- displays for each category the products with the price >8000
SELECT p.Cid, p.NameP, p.Price
FROM Category c INNER JOIN Product p ON c.Cid=p.Cid
WHERE Price>8000

-- equivalent IN
SELECT p.Cid, p.NameP, p.Price
FROM Product p
WHERE p.Price>8000 and p.Cid IN (SELECT c.Cid FROM Category c)

-- equivalent EXISTS
SELECT p.Cid, p.NameP, p.Price
FROM Product p
WHERE p.Price>8000 and EXISTS (SELECT * FROM Category c WHERE c.Cid=p.Cid)

-- equivalent FROM
SELECT A.Cid, A.NameP, A. Price
FROM (SELECT p.Cid, p.NameP, p.Price
FROM Category c INNER JOIN Product p ON c.Cid=p.Cid
WHERE Price>8000) A
```

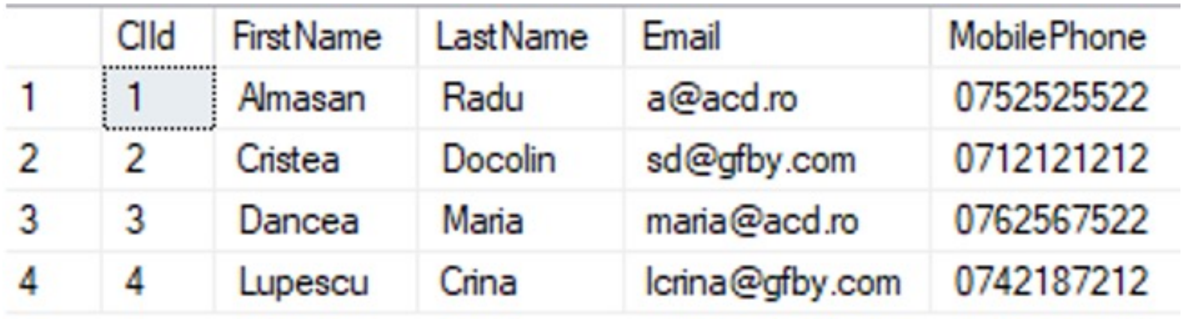


	Cid	NameP	Price
1	2	Canon d5K	8800
2	3	Nikon D850	10000
3	2	Sony Alpha9	25000
4	1	DSLR Nikon D5600	39000

SQL Queries – ANY

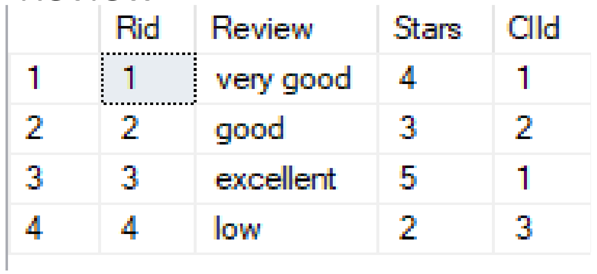
- ANY – at least one record check the condition

Client



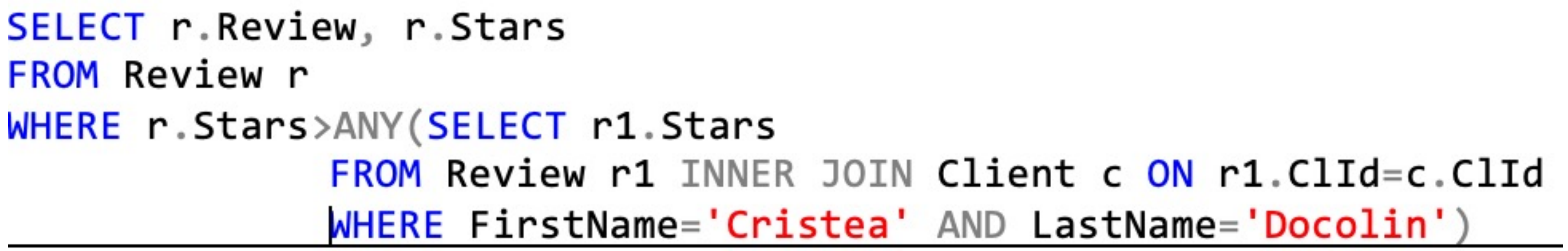
	ClId	FirstName	LastName	Email	MobilePhone
1	1	Almasan	Radu	a@acd.ro	0752525522
2	2	Cristea	Docolin	sd@gfby.com	0712121212
3	3	Dancea	Maria	maria@acd.ro	0762567522
4	4	Lupescu	Crina	lcrina@gfby.com	0742187212

Review

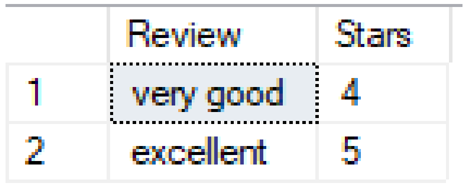


	Rid	Review	Stars	ClId
1	1	very good	4	1
2	2	good	3	2
3	3	excellent	5	1
4	4	low	2	3

○ Find the reviews that have the number of stars greater than the number of stars of a client with a specified first name and last name.



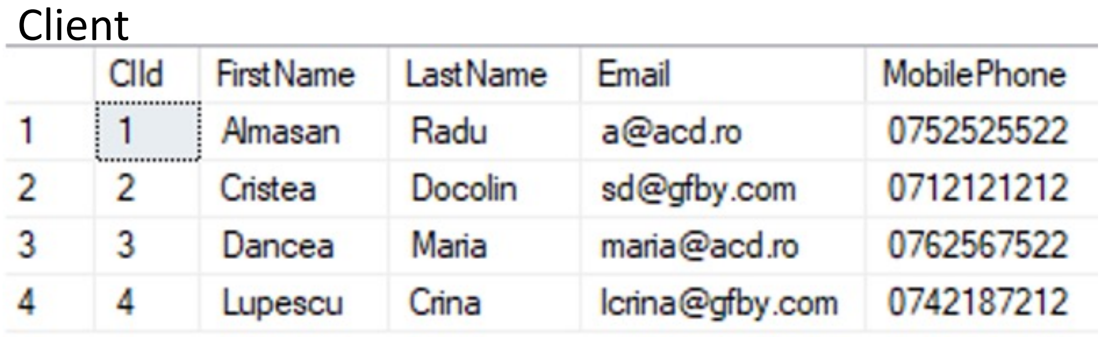
```
SELECT r.Review, r.Stars
FROM Review r
WHERE r.Stars>ANY(SELECT r1.Stars
              FROM Review r1 INNER JOIN Client c ON r1.ClId=c.ClId
              WHERE FirstName='Cristea' AND LastName='Docolin')
```



	Review	Stars
1	very good	4
2	excellent	5

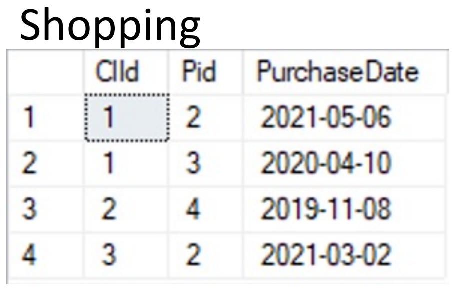
SQL Queries – ANY

- **expression = ANY(subquery)** equivalent **expression IN(subquery)**



Client

	ClId	FirstName	LastName	Email	MobilePhone
1	1	Almasan	Radu	a@acd.ro	0752525522
2	2	Cristea	Docolin	sd@gfby.com	0712121212
3	3	Dancea	Maria	maria@acd.ro	0762567522
4	4	Lupescu	Crina	lcrina@gfby.com	0742187212

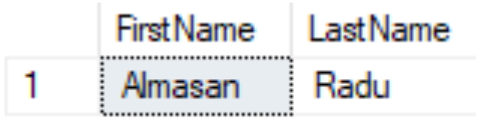


Shopping

	ClId	Pid	PurchaseDate
1	1	2	2021-05-06
2	1	3	2020-04-10
3	2	4	2019-11-08
4	3	2	2021-03-02

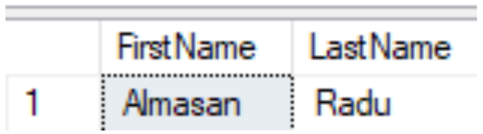
- Find the clients with a specified id that had been shopping.

```
SELECT C.FirstName, C.LastName
FROM Client C
WHERE C.ClId = ANY (SELECT S.ClId
FROM Shopping S
WHERE S.ClId=1)
```



	FirstName	LastName
1	Almasan	Radu

```
SELECT C.FirstName, C.LastName
FROM Client C
WHERE C.ClId IN (SELECT S.ClId
FROM Shopping S
WHERE S.ClId=1)
```

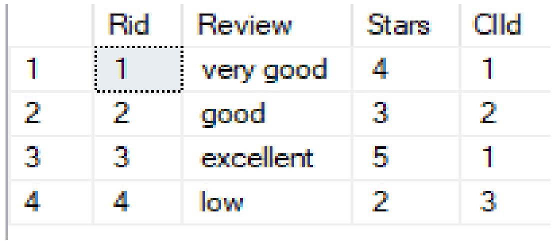


	FirstName	LastName
1	Almasan	Radu

SQL Queries – ALL

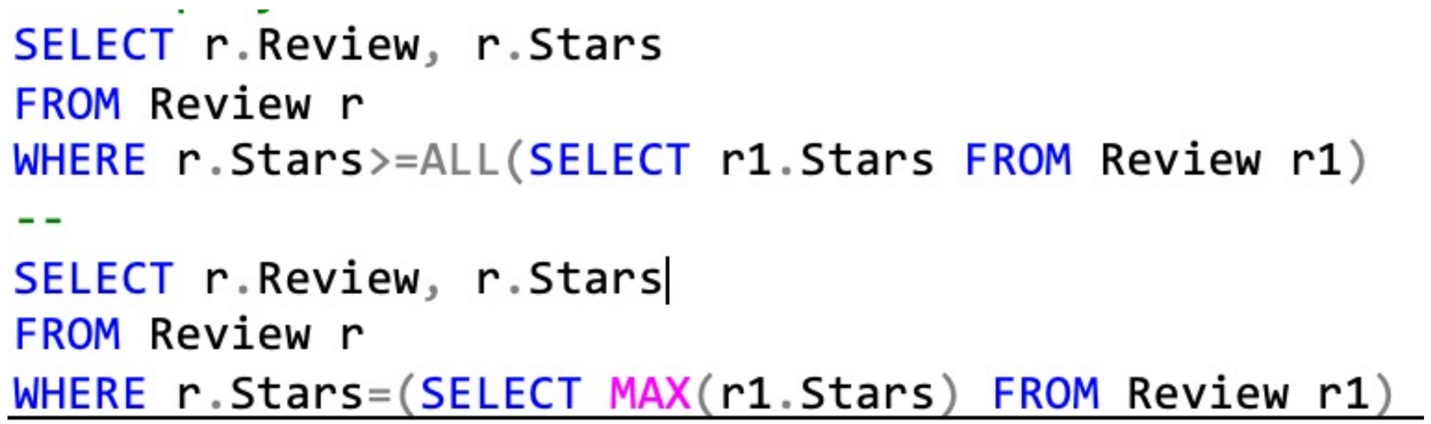
- ALL – all records check the condition

Review

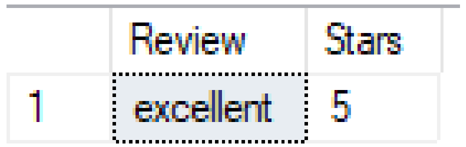


	Rid	Review	Stars	CIId
1	1	very good	4	1
2	2	good	3	2
3	3	excellent	5	1
4	4	low	2	3

- Find the reviews with the maximum number of stars.



```
SELECT r.Review, r.Stars
FROM Review r
WHERE r.Stars>=ALL(SELECT r1.Stars FROM Review r1)
--
SELECT r.Review, r.Stars
FROM Review r
WHERE r.Stars=(SELECT MAX(r1.Stars) FROM Review r1)
```

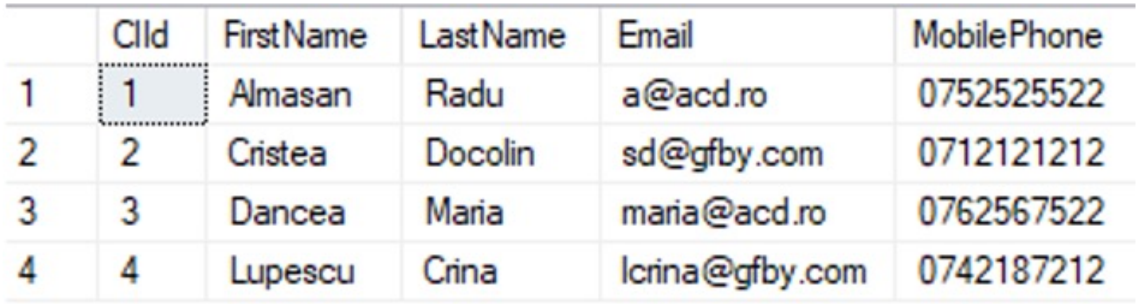


	Review	Stars
1	excellent	5

SQL Queries – ALL

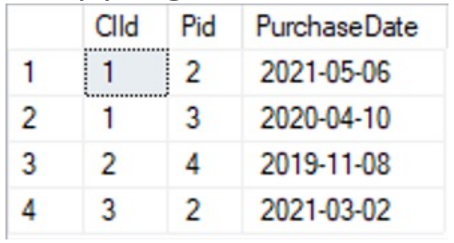
- **expression <> ALL(subquery)** equivalent **expression NOT IN(subquery)**

Client



	ClId	FirstName	LastName	Email	MobilePhone
1	1	Almasan	Radu	a@acd.ro	0752525522
2	2	Cristea	Docolin	sd@gfby.com	0712121212
3	3	Dancea	Maria	maria@acd.ro	0762567522
4	4	Lupescu	Crina	lcrina@gfby.com	0742187212

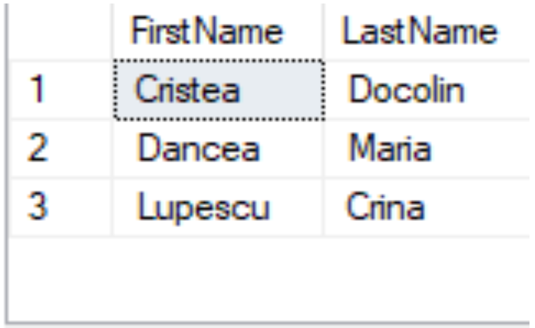
Shopping



	ClId	Pid	PurchaseDate
1	1	2	2021-05-06
2	1	3	2020-04-10
3	2	4	2019-11-08
4	3	2	2021-03-02

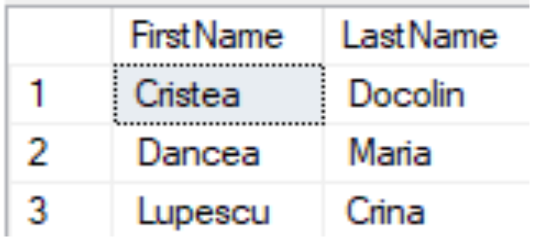
- Find the clients for which the id is not specified in Shopping.

```
SELECT C.FirstName, C.LastName
FROM Client C
WHERE C.ClId <> ALL (SELECT S.ClId
FROM Shopping S
WHERE S.ClId=1)
```



	FirstName	LastName
1	Cristea	Docolin
2	Dancea	Maria
3	Lupescu	Crina

```
SELECT C.FirstName, C.LastName
FROM Client C
WHERE C.ClId NOT IN (SELECT S.ClId
FROM Shopping S
WHERE S.ClId=1)
```



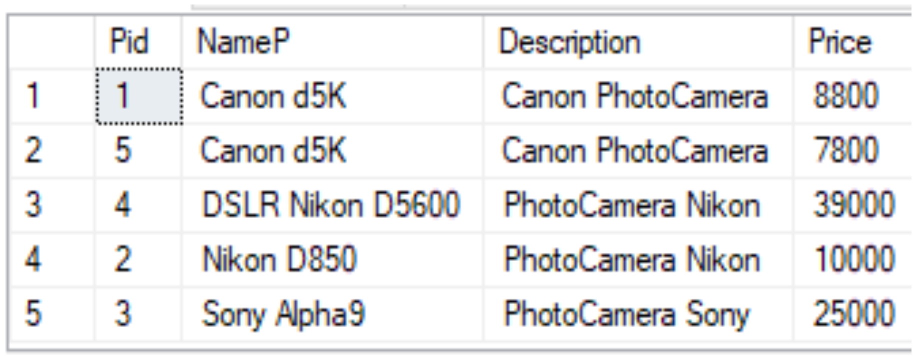
	FirstName	LastName
1	Cristea	Docolin
2	Dancea	Maria
3	Lupescu	Crina

SQL Queries – ORDER BY

- **ORDER BY** – allows to order / sort the records from the result set, after one or more fields, ASCENDING or DESCENDING

○ Sort the products by their names.

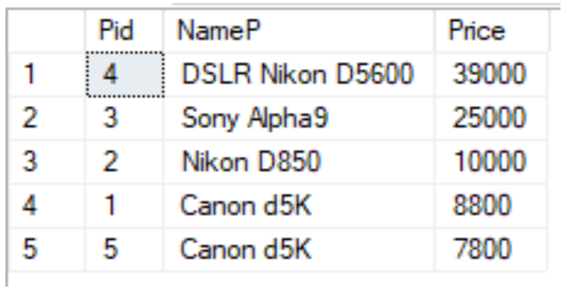
```
SELECT Pid, NameP, Description, Price
FROM Product
ORDER BY NameP ASC
-- equivalent
SELECT Pid, NameP, Description, Price
FROM Product
ORDER BY NameP
```



	Pid	NameP	Description	Price
1	1	Canon d5K	Canon PhotoCamera	8800
2	5	Canon d5K	Canon PhotoCamera	7800
3	4	DSLR Nikon D5600	PhotoCamera Nikon	39000
4	2	Nikon D850	PhotoCamera Nikon	10000
5	3	Sony Alpha9	PhotoCamera Sony	25000

○ Sort descending by price the products with the price greater than a given value.

```
SELECT Pid, NameP, Price
FROM Product
WHERE Price >500
ORDER BY Price DESC
```



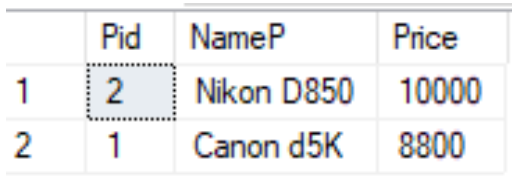
	Pid	NameP	Price
1	4	DSLR Nikon D5600	39000
2	3	Sony Alpha9	25000
3	2	Nikon D850	10000
4	1	Canon d5K	8800
5	5	Canon d5K	7800

SQL Queries – ORDER BY

- **ORDER BY** – allows to order / sort the records from the result set, after one or more fields, ASCENDING or DESCENDING

- Retrieve the products with the price between 2 given values and order descending by price and correspondgly alphabetical by name.

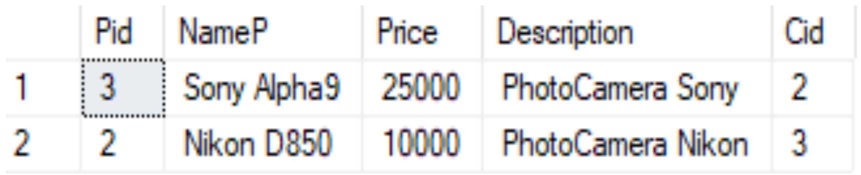
```sql
SELECT Pid, NameP, Price
FROM Product
WHERE Price BETWEEN 8000 AND 10000
ORDER BY Price DESC, NameP
```



	Pid	NameP	Price
1	2	Nikon D850	10000
2	1	Canon d5K	8800

- Find the top 25% products (from all the data) ordered by name (descending).

```sql
SELECT TOP 25 PERCENT *
FROM Product
ORDER BY NameP DESC
```



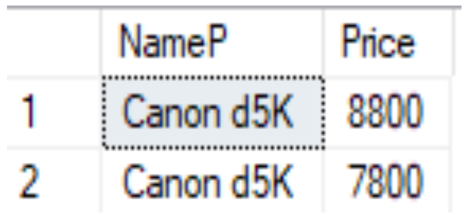
	Pid	NameP	Price	Description	Cid
1	3	Sony Alpha9	25000	PhotoCamera Sony	2
2	2	Nikon D850	10000	PhotoCamera Nikon	3

SQL Queries – SELECT … INTO

- **SELECT … INTO** – allows the saving of the result set in a table

- Find the products that have the name starting with *Canon.*

```
IF OBJECT_ID ('dbo.InsertTable', 'U') IS NOT NULL
DROP TABLE dbo.InsertTable;
GO
-- Create InsertTable
SELECT NameP, Price
INTO dbo.InsertTable
FROM Product
WHERE NameP LIKE 'Canon%'

select * from InsertTable
```

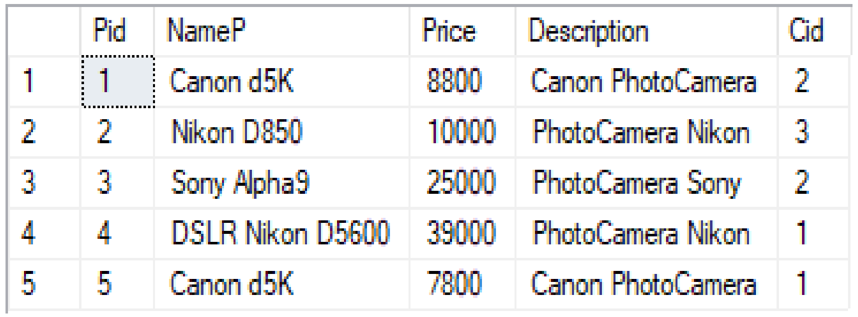


	NameP	Price
1	Canon d5K	8800
2	Canon d5K	7800

SQL Queries – UNION [ALL], INTERSECT, EXCEPT

- UNION [ALL], INTERSECT, EXCEPT – can be used with ORDER BY (at the end)

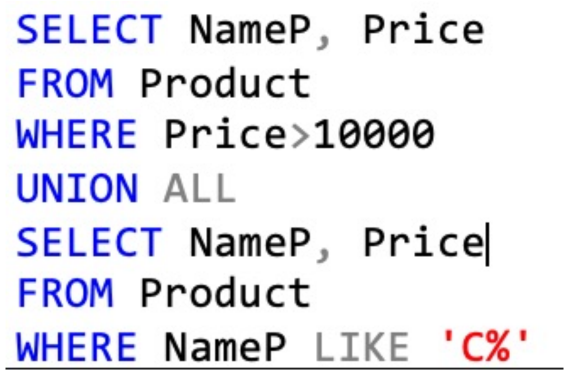
Product



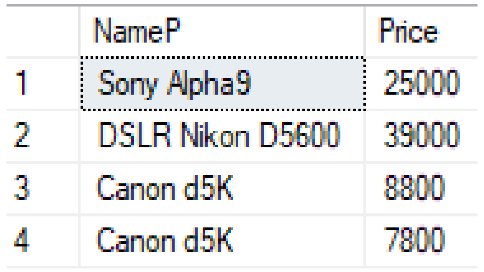
	Pid	NameP	Price	Description	Cid
1	1	Canon d5K	8800	Canon PhotoCamera	2
2	2	Nikon D850	10000	PhotoCamera Nikon	3
3	3	Sony Alpha9	25000	PhotoCamera Sony	2
4	4	DSLR Nikon D5600	39000	PhotoCamera Nikon	1
5	5	Canon d5K	7800	Canon PhotoCamera	1

- ○ **UNION** – eliminates the duplicates
- ○ **UNION ALL** – display the duplicates
- ○ works like a *WHERE* with an *OR* condition

- ○ Find the products that have the price greater than a given value OR the name starting with a given letter.



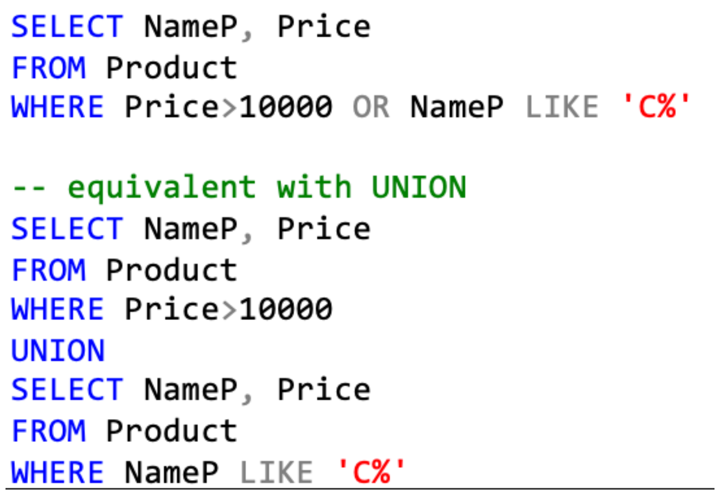
```
SELECT NameP, Price
FROM Product
WHERE Price>10000
UNION ALL
SELECT NameP, Price
FROM Product
WHERE NameP LIKE 'C%'
```



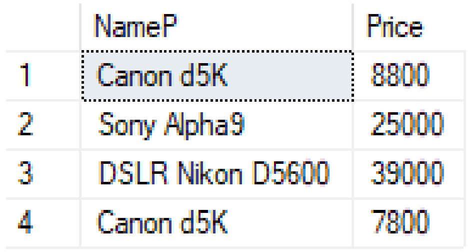
	NameP	Price
1	Sony Alpha9	25000
2	DSLR Nikon D5600	39000
3	Canon d5K	8800
4	Canon d5K	7800

SQL Queries – UNION [ALL]

- Find the products that have the price greater than a given value OR the name starting with a given letter (without duplicates).

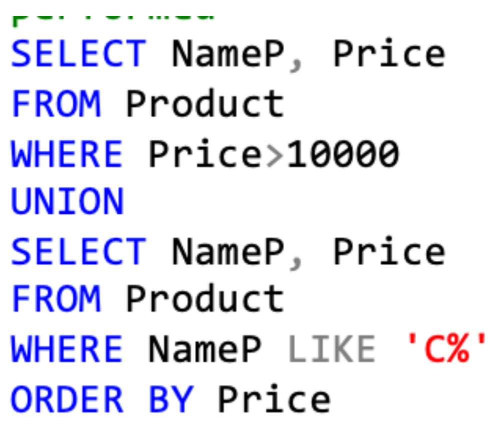


```
SELECT NameP, Price
FROM Product
WHERE Price>10000 OR NameP LIKE 'C%'

-- equivalent with UNION
SELECT NameP, Price
FROM Product
WHERE Price>10000
UNION
SELECT NameP, Price
FROM Product
WHERE NameP LIKE 'C%'
```

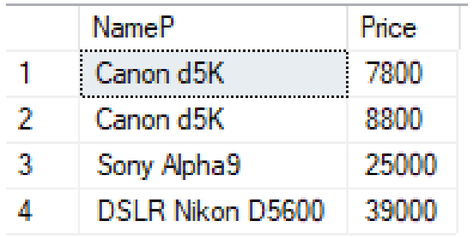


	NameP	Price
1	Canon d5K	8800
2	Sony Alpha9	25000
3	DSLR Nikon D5600	39000
4	Canon d5K	7800

- Find the products that have the price greater than a given value OR the name starting with a given letter (without duplicates).



```
SELECT NameP, Price
FROM Product
WHERE Price>10000
UNION
SELECT NameP, Price
FROM Product
WHERE NameP LIKE 'C%'
ORDER BY Price
```



	NameP	Price
1	Canon d5K	7800
2	Canon d5K	8800
3	Sony Alpha9	25000
4	DSLR Nikon D5600	39000

SQL Queries - INTERSECT

- works like a *WHERE* with an *AND* condition
- intersection queries can be expressed with IN

○ Find the products that have the price greater than a given value AND the name starting with a given letter.

```
SELECT NameP, Price
FROM Product
WHERE Price>10000 AND NameP LIKE 'C%'
```

NameP	Price

```
-- equivalent with INTERSECT
SELECT NameP, Price
FROM Product
WHERE Price>10000
INTERSECT
SELECT NameP, Price
FROM Product
WHERE NameP LIKE 'C%'
```

NameP	Price

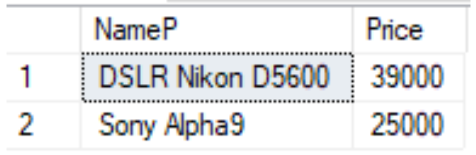
```
-- equivalent with IN
SELECT P.NameP, P.Price
FROM Product P
WHERE P.Price>10000 AND P.NameP IN (SELECT P1.NameP
                                  FROM Product P1 WHERE P1.NameP LIKE 'C%')
```

SQL Queries - EXCEPT

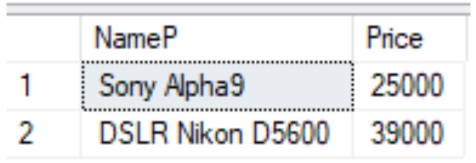
- works like a *set - difference* – first condition fulfilled and the second condition not fulfilled
- set-difference queries can be expressed with NOT IN

○ Find the products that have the price greater than a given value BUT WITHOUT the name starting with a given letter.

```
SELECT NameP, Price
FROM Product
WHERE Price>10000
EXCEPT
SELECT NameP, Price
FROM Product
WHERE NameP LIKE 'C%'
-- equivalent with NOT IN
SELECT P.NameP, P.Price
FROM Product P
WHERE P.Price>10000 AND P.NameP NOT IN (SELECT P1.NameP
                                  FROM Product P1 WHERE P1.NameP LIKE 'C%')
```



	NameP	Price
1	DSLR Nikon D5600	39000
2	Sony Alpha9	25000



	NameP	Price
1	Sony Alpha9	25000
2	DSLR Nikon D5600	39000

SQL Queries – UNION, INTERSECT, EXCEPT

Common mistaces (ERRORS):

- ORDER BY can be used only in the end of the query.

```sql
SELECT NameP, Price
FROM Product
WHERE Price>10000
ORDER BY Price
UNION
SELECT NameP, Price
FROM Product
WHERE NameP LIKE 'C%'
```

```
Msg 156, Level 15, State 1, Line 408
Incorrect syntax near the keyword 'UNION'.
```

- The fields from the both SELECT's should have the same number, type (and order).

```sql
SELECT NameP, Price
FROM Product
WHERE Price>10000
UNION
SELECT NameP
FROM Product
WHERE NameP LIKE 'C%'
ORDER BY Price
```

```
Msg 205, Level 16, State 1, Line 424
All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.
```

```sql
SELECT NameP, Price
FROM Product
WHERE Price>10000
UNION
SELECT Price, NameP
FROM Product
WHERE NameP LIKE 'C%'
ORDER BY Price
```

```
Msg 245, Level 16, State 1, Line 433
Conversion failed when converting the varchar value 'Sony Alpha9' to data type int.
```

SQL Queries – NULL value

- In some circumstances, the particular values of the attributes / fields can be *unknown* or *unusuable* temporary – SQL allows to use this special value ***NULL***.

- By using NULL:
  - It is necesarry to implement a logic with 3 values: TRUE, FALSE, NULL
  (e.g. if the value of the field *Price* is NULL, then a condition like *Price<value* is going to be evaluted with FALSE)
  - It is necessary to add a special operator : IS NULL / IS NOT NULL

SQL Queries – GROUP BY, HAVING

```
SELECT [DISTINCT] select_list
FROM from_list
WHERE qualification
GROUP BY group_by_list
HAVING group_qualification
```

- Each tuple from the result set corresponds to a group and all the attributes will have a value per group

- **GROUP BY, HAVING** clauses – are optional
- **GROUP BY** clause – list of (expressions involving) columns used for grouping;
- - a collection of rows with identical values for the columns in ***group_by_list***
- every row in the result set of the query correponds to a group
- **HAVING** clause – group qualification conditions
- AGGREGATION OPERATORS : **COUNT, AVG, SUM, MIN, MAX**
- **COUNT(*), COUNT([DISTINCT] A), SUM([DISTINCT] A), AVG([DISTINCT] A), MAX(A), MIN(A),** where A is an attribute

SQL Queries – GROUP BY, HAVING

SELECT [DISTINCT] select_list
FROM from_list
WHERE qualification
GROUP BY group_by_list
HAVING group_qualification

- **select_list** - columns from here must appear in *group_by_list*
  - the terms have the form: ***aggregation_operator(column_name) [AS NewName]*** , where NewName assigns a name to the column in the result set / table
- **group_qualification** - expressions with a single value / group
  - a column in ***group_qualification*** appears in ***group_by_list*** or as an argument to an aggregation operator
  - contains condition(s) on the aggregate functions
- the records that meet ***qualification*** are partitioned into groups based on the values of the columns in ***group_by_list***
- a result row is generated for every group that meets ***group_qualification***

SQL Queries – example

Example on the schema:

- Category [Cid, Name]
- Product [Pid, NameP, Price, Description, Cid]

- Display the average price for each category.

```
SELECT C.Cid,  AVG(Price) AS average_price
FROM Category C,  Product P
WHERE C.Cid=P.Cid
GROUP BY C.Cid
```

Category

Cid	Name
1	Professional
2	Custom
3	Usual

Product

Pid	NameP	Price	Description	Cid
1	Canon d5K	8800	Canon PhotoCamera	2
2	Nikon D850	10000	PhotoCamera Nikon	3
3	Sony Alpha 9	25000	PhotoCamera Sony	2
4	DSLR Nikon D5600	39000	PhotoCamera Nikon	1
5	Canon d5K	7800	Canon PhotoCamera	1

SQL Queries – example

- After computing the condition C.Cid=P.Cid:

Cid	Name	Pid	NameP	Price	Description	Cid
2	Custom	1	Canon d5K	8800	Canon PhotoCamera	2
3	Usual	2	Nikon D850	10000	PhotoCamera Nikon	3
2	Custom	3	Sony Alpha 9	25000	PhotoCamera Sony	2
1	Professional	4	DSLR Nikon D5600	39000	PhotoCamera Nikon	1
1	Professional	5	Canon d5K	7800	Canon PhotoCamera	1

- Apply SELECT C.Cid, AVG(Price) AS average_price … GROUP BY C.Cid

Cid	Name	Pid	NameP	Price	Description	Cid
2	Custom	1	Canon d5K	8800	Canon PhotoCamera	2
3	Usual	2	Nikon D850	10000	PhotoCamera Nikon	3
2	Custom	3	Sony Alpha 9	25000	PhotoCamera Sony	2
1	Professional	4	DSLR Nikon D5600	39000	PhotoCamera Nikon	1
1	Professional	5	Canon d5K	7800	Canon PhotoCamera	1

SQL Queries – example

- Apply SELECT C.Cid, AVG(Price) AS average_price … GROUP BY C.Cid

Cid	average_price
3	10000
2	(8800+25000)/2
1	(39000+7800)/2

- The query result is:

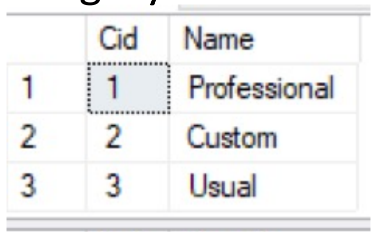
Cid	average_price
1	23400
2	16900
3	10000

Display the average price for each category.

```
SELECT C.Cid,  AVG(Price) AS average_price
FROM Category C,  Product P
WHERE C.Cid=P.Cid
GROUP BY C.Cid
```

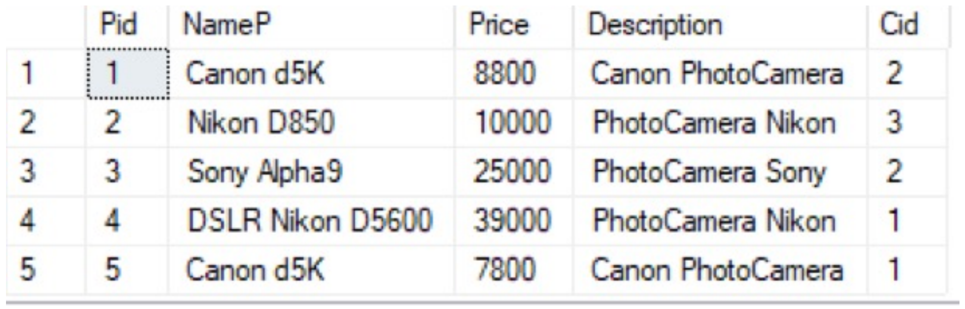
SQL Queries – GROUP BY

Category



	Cid	Name
1	1	Professional
2	2	Custom
3	3	Usual

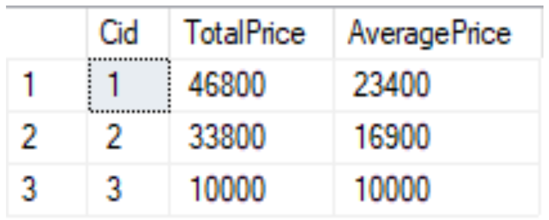
Product



	Pid	NameP	Price	Description	Cid
1	1	Canon d5K	8800	Canon PhotoCamera	2
2	2	Nikon D850	10000	PhotoCamera Nikon	3
3	3	Sony Alpha9	25000	PhotoCamera Sony	2
4	4	DSLR Nikon D5600	39000	PhotoCamera Nikon	1
5	5	Canon d5K	7800	Canon PhotoCamera	1

- Find for each category, the total price and the average price of the products included.

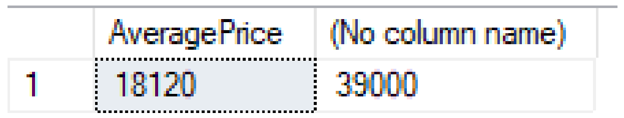
```sql
SELECT C.Cid, SUM(Price) AS TotalPrice, AVG(Price) AS AveragePrice
FROM Category C, Product P
WHERE C.Cid=P.Cid
GROUP BY C.Cid
```



	Cid	TotalPrice	AveragePrice
1	1	46800	23400
2	2	33800	16900
3	3	10000	10000

- Find the average price and the maximum price.

```sql
SELECT AVG(Price) AS AveragePrice, MAX(Price)
FROM Product
```

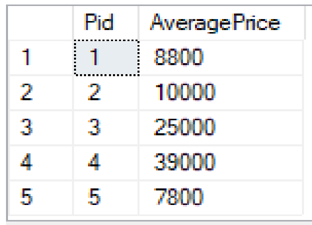


	AveragePrice	(No column name)
1	18120	39000

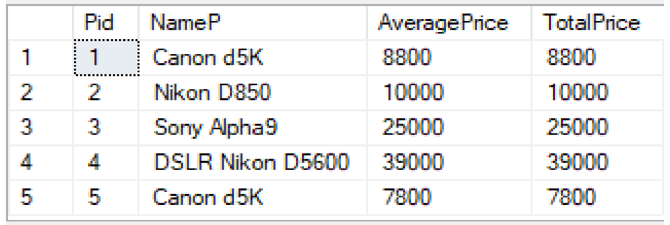
SQL Queries – GROUP BY, HAVING

- Find for each product, the average price and the total price.

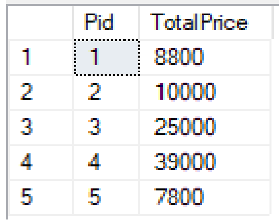
```
SELECT Pid, AVG(Price) AS AveragePrice
FROM Product
GROUP BY Pid

SELECT Pid, NameP, AVG(Price) AS AveragePrice, SUM(Price) AS TotalPrice
FROM Product
GROUP BY Pid, NameP
ORDER BY Pid
|
SELECT Pid, SUM(Price) AS TotalPrice
FROM Product
GROUP BY Pid, NameP
```



	Pid	AveragePrice
1	1	8800
2	2	10000
3	3	25000
4	4	39000
5	5	7800



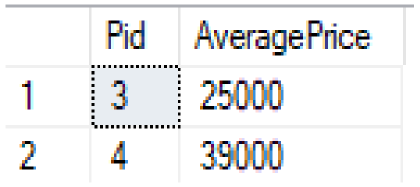
	Pid	NameP	AveragePrice	TotalPrice
1	1	Canon d5K	8800	8800
2	2	Nikon D850	10000	10000
3	3	Sony Alpha9	25000	25000
4	4	DSLR Nikon D5600	39000	39000
5	5	Canon d5K	7800	7800



	Pid	TotalPrice
1	1	8800
2	2	10000
3	3	25000
4	4	39000
5	5	7800

- Find for each product, with the price greater than a given value, the average price (conditions on the average and the sum of the price).

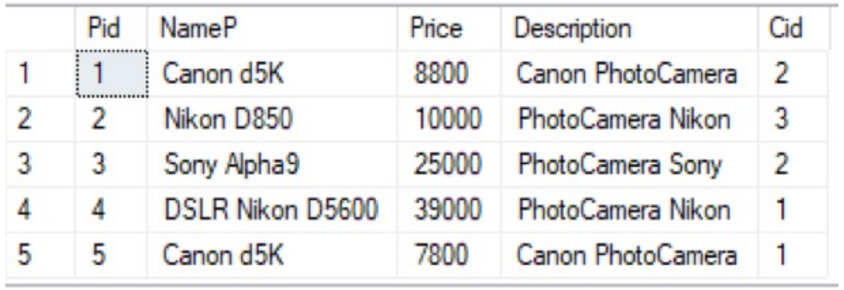
```
SELECT Pid, AVG(Price) AS AveragePrice
FROM Product
WHERE Price>9000 -- condition(s) for the fields from the table(s)
GROUP BY Pid
HAVING  AVG(Price)>20000  OR  SUM(Price)  BETWEEN  12000  AND  1200000  --
Condition(s) for the aggregate functions
ORDER BY Pid
```



	Pid	AveragePrice
1	3	25000
2	4	39000

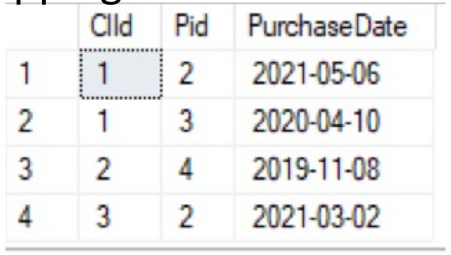
SQL Queries – GROUP BY

Product



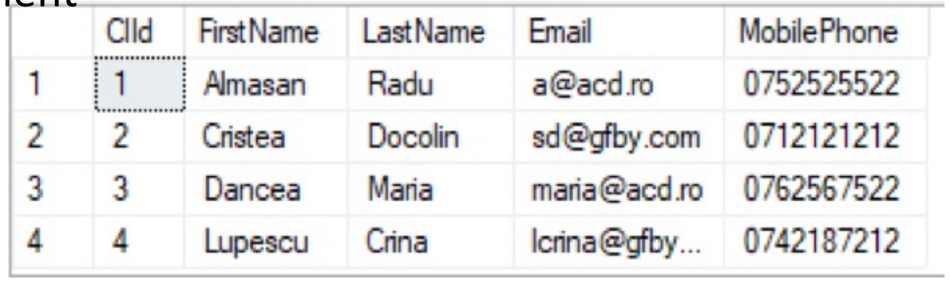
	Pid	NameP	Price	Description	Cid
1	1	Canon d5K	8800	Canon PhotoCamera	2
2	2	Nikon D850	10000	PhotoCamera Nikon	3
3	3	Sony Alpha9	25000	PhotoCamera Sony	2
4	4	DSLR Nikon D5600	39000	PhotoCamera Nikon	1
5	5	Canon d5K	7800	Canon PhotoCamera	1

Shopping



	ClId	Pid	PurchaseDate
1	1	2	2021-05-06
2	1	3	2020-04-10
3	2	4	2019-11-08
4	3	2	2021-03-02

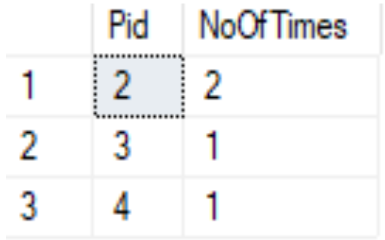
Client



	ClId	FirstName	LastName	Email	MobilePhone
1	1	Almasan	Radu	a@acd.ro	0752525522
2	2	Cristea	Docolin	sd@gfby.com	0712121212
3	3	Dancea	Maria	maria@acd.ro	0762567522
4	4	Lupescu	Crina	lcrina@gfby...	0742187212

- Find for each product, how many times it was bought.

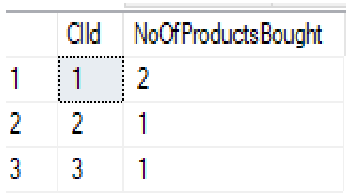
```
SELECT P.Pid, COUNT(*) AS NoOfTimes
FROM Product P, Shopping S
WHERE P.Pid=S.Pid AND Price>1000
GROUP BY P.Pid
```



	Pid	NoOfTimes
1	2	2
2	3	1
3	4	1

- Find the number of products bought by each client.

```
SELECT C.ClId, COUNT(*) NoOfProductsBought
FROM Client C, Shopping S
WHERE C.ClId=S.ClId
GROUP BY C.ClId
```

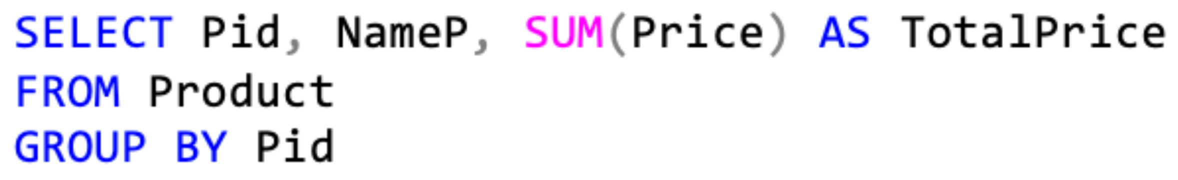


	ClId	NoOfProductsBought
1	1	2
2	2	1
3	3	1

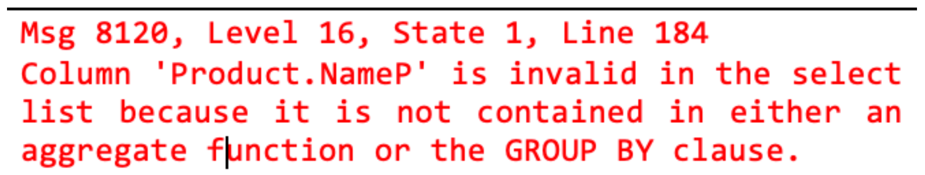
SQL Queries – GROUP BY, HAVING

Common mistaces (ERRORS):

- Number of columns in SELECT and GROUP BY should be the same and with the same type.

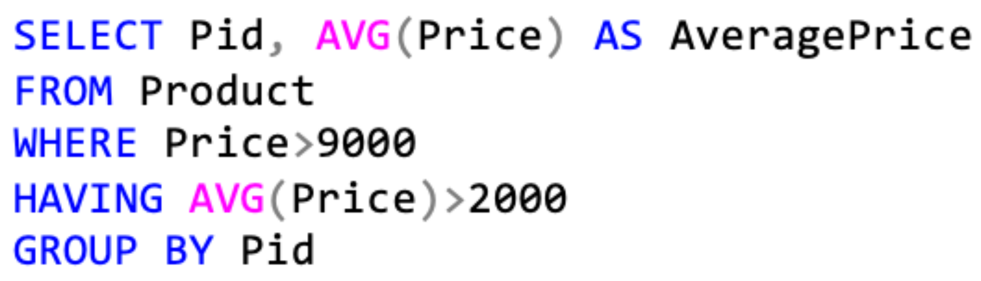


```sql
SELECT Pid, NameP, SUM(Price) AS TotalPrice
FROM Product
GROUP BY Pid
```

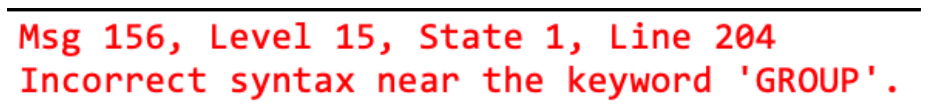


```
Msg 8120, Level 16, State 1, Line 184
Column 'Product.NameP' is invalid in the select
list because it is not contained in either an
aggregate function or the GROUP BY clause.
```

- HAVING clause follows the GROUP BY (not before).



```sql
SELECT Pid, AVG(Price) AS AveragePrice
FROM Product
WHERE Price>9000
HAVING AVG(Price)>2000
GROUP BY Pid
```

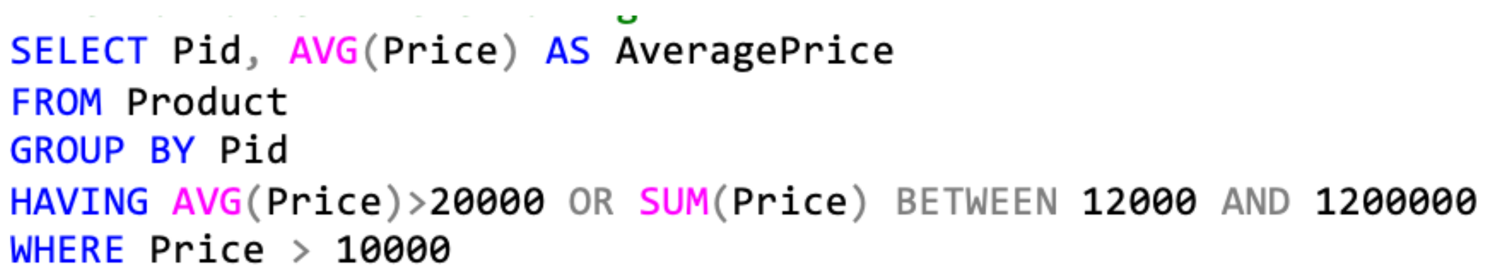


```
Msg 156, Level 15, State 1, Line 204
Incorrect syntax near the keyword 'GROUP'.
```

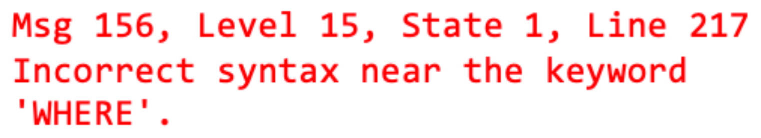
SQL Queries – GROUP BY, HAVING

Common mistaces (ERRORS):

- The order of the clauses in the SELECT statement is mandatory.

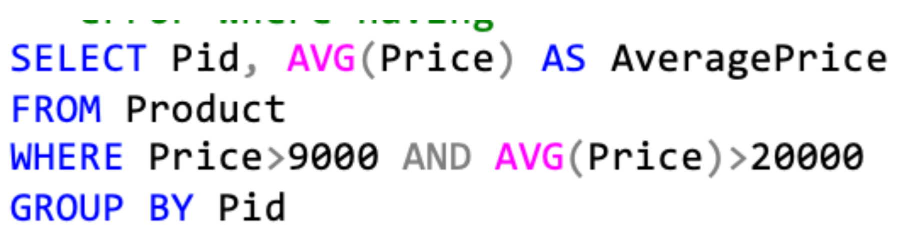


```sql
SELECT Pid, AVG(Price) AS AveragePrice
FROM Product
GROUP BY Pid
HAVING AVG(Price)>20000 OR SUM(Price) BETWEEN 12000 AND 1200000
WHERE Price > 10000
```

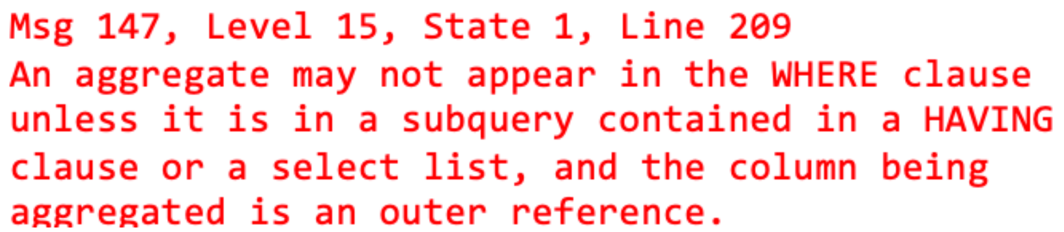


```
Msg 156, Level 15, State 1, Line 217
Incorrect syntax near the keyword
'WHERE'.
```

- Conditions on aggregate functions can be put it ONLY in the HAVING clause.



```sql
SELECT Pid, AVG(Price) AS AveragePrice
FROM Product
WHERE Price>9000 AND AVG(Price)>20000
GROUP BY Pid
```



```
Msg 147, Level 15, State 1, Line 209
An aggregate may not appear in the WHERE clause
unless it is in a subquery contained in a HAVING
clause or a select list, and the column being
aggregated is an outer reference.
```

SQL Queries - SELECT statement

SELECT [ALL/ DISTINCT / TOP n [PERCENT]] * / column(s)_name / expressions

FROM table_name1 [ALIAS], table_name2 [ALIAS], … / view / (nested) select_statement / join_expression

[WHERE qualification]

[GROUP BY group_by_list]

[HAVING group_qualification]

[UNION [ALL] / INTERSECT / EXCEPT] SELECT_Statement]

[ORDER BY column_name1 [ASC / DESC], [column_name2 [ASC / DESC], …]]

- SELECT statement result-set: a relation (table).
- WHERE qualification – can contain filter (expressions with relational operators, [NOT] LIKE, IS [NOT] NULL, [NOT] BETWEEN min_value AND max_value, [NOT] IN (value [, value] …) / (subquery), [NOT] EXISTS (subquery), ALL / ANY (subquery)) and join conditions

References:

- C.J. Date, *An Introduction to Database Systems (8th Edition)*, Addison-Wesley, 2003.
- H. Garcia-Molina, J. Ullman, J. Widom, *Database Systems: The Complete Book, Prentice Hall Press*, 2008.
- G. Hansen, J. Hansen, *Database Management And Design (2nd Edition),* Prentice Hall, 1996.
- R. Ramakrishnan, J. Gehrke, *Database Management Systems*, McGraw- Hill, 2007. http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html
- R. Ramakrishnan, J. Gehrke, *Database Management Systems (2nd Edition),* McGraw-Hill, 2000.
- A. Silberschatz, H. Korth, S. Sudarshan, *Database System Concepts*, McGraw-Hill, 2010. http://codex.cs.yale.edu/avi/db-book/
- L. Țâmbulea, *Curs Baze de date*, Facultatea de Matematică și Informatică, UBB, 2013-2014.
- J. Ullman, J. Widom, *A First Course in Database Systems*, http://infolab.stanford.edu/~ullman/fcdb.html