**Lab 2**

Group 1

Lab team: Ievgenii Nudga, Aryan Sultan

Instructor: Florian Ocker

HAW Hamburg Data Base

**Assignment 8: Functional Dependencies and Normalization**

**A) The (full) Functional Dependencies**

lect\_id -> lect\_name

prof\_id -> prof\_name

note\_id -> price, quantity

**B) Primary keys for the table**

lect\_id, prof\_id, note\_id

**C) Transforming the schema to 2NF**

1NF

sells (lect\_id, lect\_name, prof\_id, prof\_name, note\_id, price, quantity)

2NF

lect\_id (lect\_name, prof\_id, prof\_name, note\_id, price, quantity)

prof\_id (prof\_name, lect\_id, lect\_name, note\_id)

note\_id (lect\_id, lect\_name, prof\_id, prof\_name, price)

**Assignment 9: DML**

**Script:**

DELETE FROM Person WHERE Name1 = 'Walt' OR Name1 = 'Ellen' OR Name1 = 'Bob';

INSERT INTO Person VALUES('Alice', TO\_DATE( '02-02-1955', 'DD-MM-YYYY' ), null, null, null);

INSERT INTO Person VALUES('Albert', TO\_DATE( '03-03-1956', 'DD-MM-YYYY' ), null, null, null);

INSERT INTO Person VALUES('Trent', TO\_DATE('04-07-1986', 'DD-MM-YYYY' ), null, 'Albert', 'Alice');

UPDATE Person SET Father = 'Trent' WHERE Name1 = 'Jane' OR Name1 = 'Joe' ;

**Output:**

3 rows deleted.

1 row inserted.

1 row inserted.

1 row inserted.

2 rows updated.

**Assignment 10: Queries**

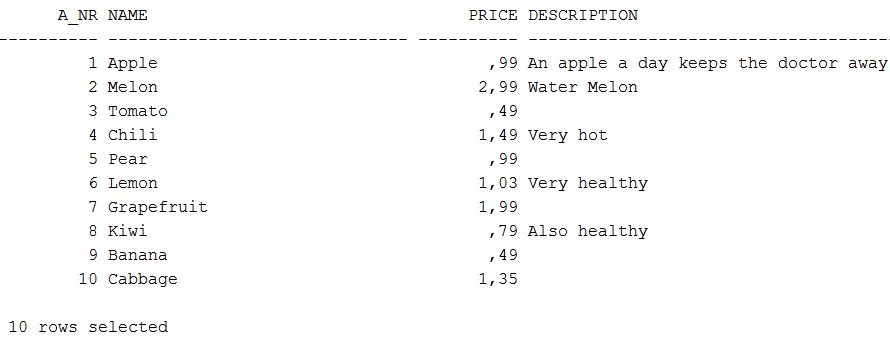
1. **Display all articles:**

SELECT \*FROM article;

**it would be nice to know ways to manipulate with tables, like adjusting words on each column on the left side, also I wanted to get 0.49, instead of ,49**

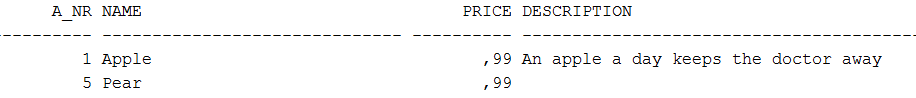
**so far I found about SET LINESIZE 2000;**

**I googled a bit more, but the code is different and maybe I should look into program settings and try finding some tweaks?**



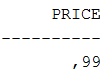
1. **List articles with price 0.99**

SELECT \* FROM article WHERE price = 0.99;



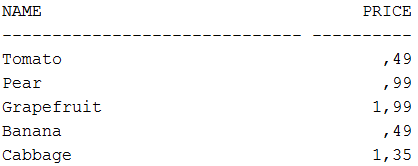
1. **Price of the article Apple**

SELECT price FROM article WHERE name = ‘Apple’;



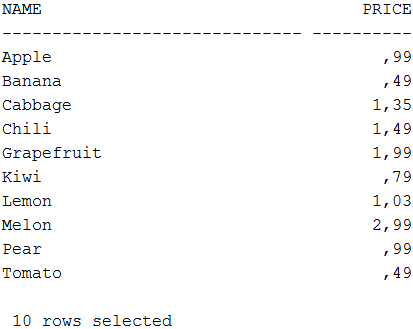
1. **List of articles wit NULL description**

SELECT name, price FROM article WHERE description IS NULL;



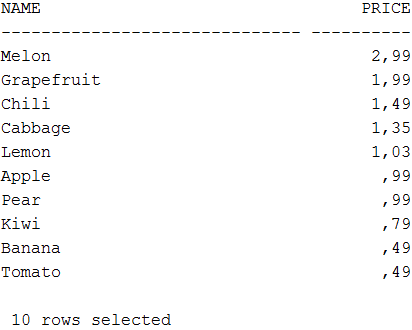
1. **List of articles in alphabetical order**

SELECT name, price FROM article ORDER BY name ASC;



1. **List of articles (most expensive first)**

SELECT name, price FROM article ORDER BY price DESC;



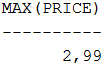
1. **The average price of articles**

SELECT AVG (price) FROM article;



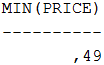
1. **Highest price of articles**

SELECT MAX (price) FROM article;



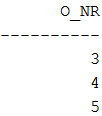
1. **Lowest price of articles**

SELECT MIN (price) FROM article;



1. **List of order numbers (o\_nr) of all Ringo’s orders**

SELECT orders.o\_nr FROM customer, orders WHERE customer.name = 'Ringo' AND customer.c\_id = orders.c\_id ORDER BY orders.o\_nr ASC;

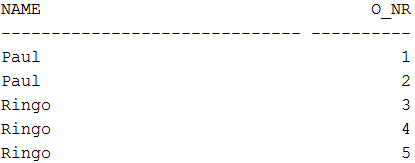


1. **List of customer names and their order numbers**

SELECT customer.name, orders.o\_nr FROM customer, orders WHERE customer.c\_id = orders.c\_id ORDER BY orders.o\_nr ASC;

or

SELECT customer.name, orders.o\_nr FROM customer INNER JOIN orders ON customer.c\_id = orders.c\_id ORDER BY orders.o\_nr ASC;



1. **List of ALL customer names and corresponding order numbers (the difference is FULL OUTER JOIN so that we can also get names without any data (missing values o\_nr filled with NULLs)**

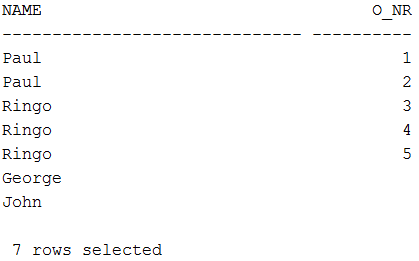
**Sometimes program shows the amount of rows selected, sometimes it doesn’t, should I be worried about that?**

SELECT customer.name, orders.o\_nr FROM customer FULL OUTER JOIN orders ON customer.c\_id = orders.c\_id ORDER BY orders.o\_nr ASC;

but if we use this

SELECT customer.name, orders.o\_nr FROM customer, orders WHERE customer.c\_id = orders.c\_id ORDER BY orders.o\_nr ASC;

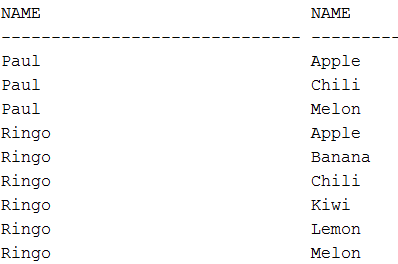
then there wouldn’t be names with missing values o\_nr



1. **Articles that has been ordered by every single customer**

SELECT DISTINCT customer.name, article.name FROM customer, article, order\_item, orders

WHERE customer.c\_id = orders.c\_id AND orders.o\_nr = order\_item.o\_nr AND article.a\_nr = order\_item.a\_nr ORDER BY customer.name, article.name ASC;

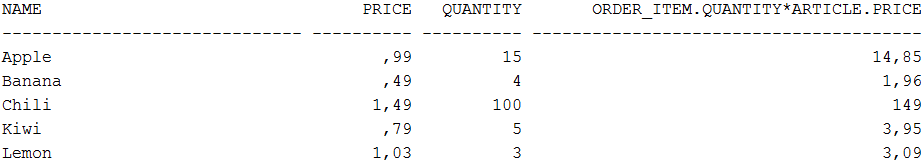


1. **Summary for order 5**

SELECT DISTINCT article.name, article.price, order\_item.quantity, order\_item.quantity \* article.price

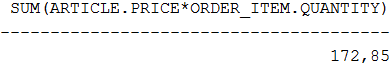
FROM article, order\_item WHERE order\_item.o\_nr = 5 AND article.a\_nr = order\_item.a\_nr

ORDER BY article.name ASC;



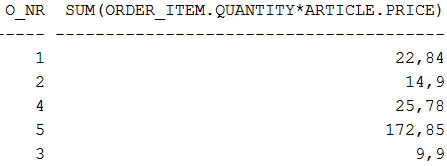
1. **The sum price of order 5**

SELECT SUM (article.price \* order\_item.quantity) FROM article, order\_item WHERE order\_item.o\_nr = 5 AND article.a\_nr = order\_item.a\_nr ORDER BY article.name ASC;



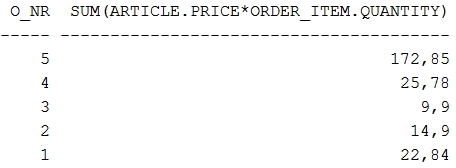
1. **Summary of all orders (o\_nr and total price)**

SELECT order\_item.o\_nr, SUM (article.price \* order\_item.quantity) FROM article, order\_item, orders WHERE orders.o\_nr = order\_item.o\_nr AND article.a\_nr = order\_item.a\_nr GROUP BY order\_item.o\_nr;



**Experiment**

SELECT order\_item.o\_nr, SUM (article.price \* order\_item.quantity) FROM article, order\_item, orders WHERE orders.o\_nr = order\_item.o\_nr AND article.a\_nr = order\_item.a\_nr GROUP BY order\_item.o\_nr ORDER BY order\_item.o\_nr DESC;



1. **Summary of all orders (o\_nr, customer name and total price)**

SELECT order\_item.o\_nr, customer.name, SUM (article.price \* order\_item.quantity)

FROM article, order\_item, orders, customer

WHERE orders.o\_nr = order\_item.o\_nr AND article.a\_nr = order\_item.a\_nr AND customer.c\_id = orders.c\_id GROUP BY order\_item.o\_nr, customer.name ORDER BY order\_item.o\_nr ASC;

