

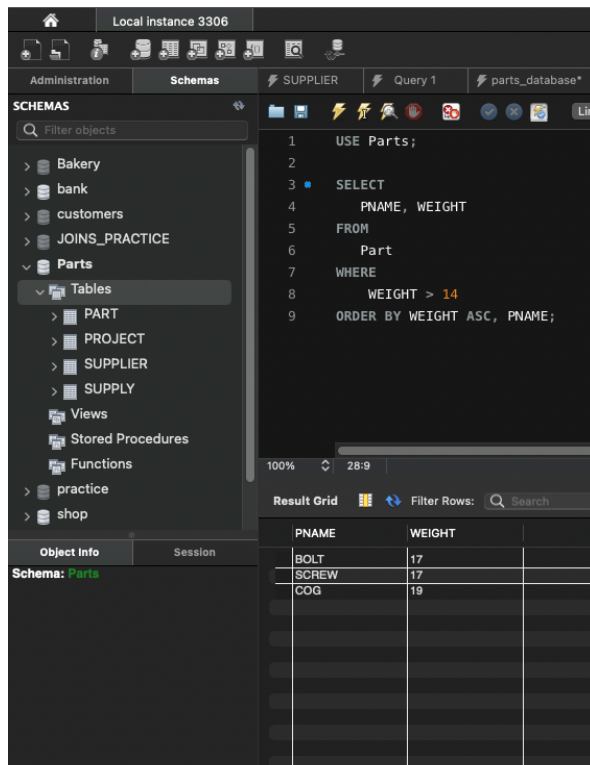
CFG-NANODEGREE
FULL-STACK
SUBMITTED TO:
INSTRUCTOR: MR. SCOTT ADAMS

PAULA MANESE-FULL STACK CLASS

HOMEWORK WEEK 1 - SQL

USE PARTS DB TO WRITE THE FOLLOWING QUERIES

1. Find the name of each part where the weight is more than 14.



ANSWER:

USE Parts;

```
SELECT  
  PNAME, WEIGHT  
FROM  
  Part  
WHERE  
  WEIGHT > 14  
ORDER BY WEIGHT ASC, PNAME;
```

2. Find all **unique** supplier(s) where their status is equal to 20.

The screenshot shows a database management interface with a dark theme. On the left, a 'SCHEMAS' panel lists various databases including Bakery, bank, customers, JOINS_PRACTICE, Parts, practice, shop, sys, and tutorial. The 'Parts' database is selected. The main area displays a SQL query in a text editor:

```
1 USE Parts;
2
3 SELECT DISTINCT
4     STATUS, SNAME
5 FROM SUPPLIER
6 WHERE STATUS = '20';
```

Below the query editor, a 'Result Grid' shows the results of the query. The grid has two columns: 'STATUS' and 'SNAME'. The results are as follows:

STATUS	SNAME
20	SMITH
20	CLARK

ANSWER:

```
USE Parts;
SELECT DISTINCT
    STATUS, SNAME
FROM SUPPLIER
WHERE STATUS = '20';
```

TASK 2

USE SHOP SALES DB TO WRITE THE FOLLOWING QUERIES

1. Find out how many sales (amount) were recorded each week, per day (where available)

○ **This would look like:**

Week 1, Tuesday, £x

Week 1, Wednesday, £x

Week 2, Monday, £x

Week 2, Friday, £x

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Schemas' tree with 'shop' expanded, showing 'Tables' and 'SALES1'. The right pane shows a query window with the following SQL code:

```
1 USE shop;
2
3 SELECT CONCAT('Week ',Week) as 'Week', Day, CONCAT('£',sum(SalesAmount)) As Sales
4 FROM SALES1
5 GROUP BY Month,Week,Day
6 ORDER BY Week, Day ASC;
7
8
9
```

Below the query window, the 'Result Grid' shows the following data:

Week	Day	Sales
Week 1	Saturday	£43.11
Week 1	Tuesday	£44.27
Week 2	Monday	£56.25
Week 3	Tuesday	£9.99
Week 4	Monday	£77.00
Week 4	Wednesday	£86.81
Week 5	Monday	£98.42
Week 5	Saturday	£79.90
Week 5	Tuesday	£74.32
Week 6	Friday	£74.02

ANSWER:

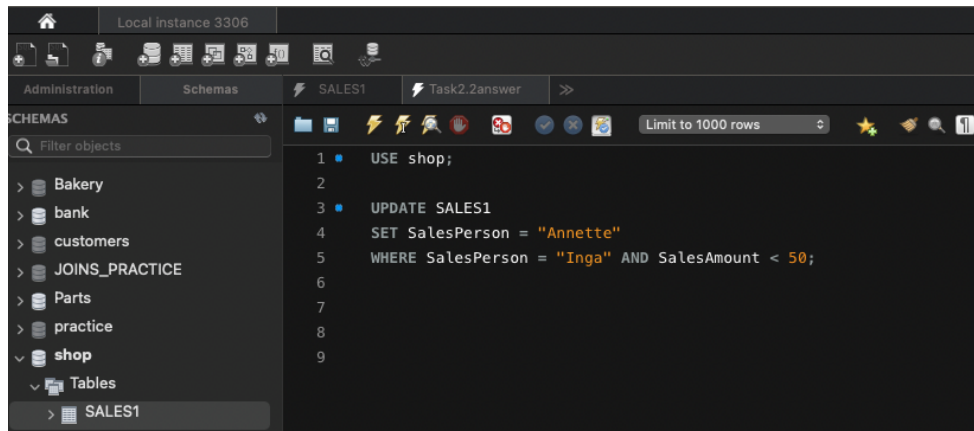
USE shop;

SELECT CONCAT('Week ',Week) as 'Week', Day, CONCAT('£',sum(SalesAmount)) As Sales
FROM SALES1

GROUP BY Month,Week,Day

ORDER BY Week, Day ASC;

2. Change the name of salesperson Inga to be Annette instead, but only where Ignas Sales are <50.



100% 1:1

Result Grid Filter Rows: Search Export:

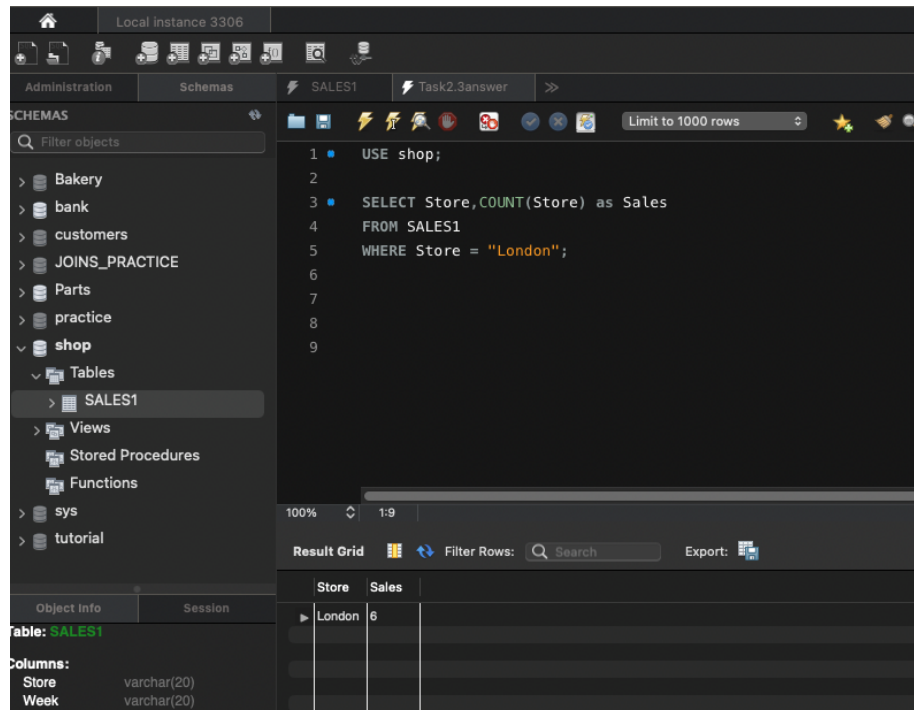
	Store	Week	Day	SalesPerson	SalesAmount	Month	
	London	2	Monday	Frank	56.25	May	
	London	5	Tuesday	Frank	74.32	Sep	
	London	5	Monday	Bill	98.42	Sep	
	London	5	Saturday	Bill	73.90	Dec	
	London	1	Tuesday	Josie	44.27	Sep	
	Dusseldorf	4	Monday	Manfred	77.00	Jul	
▶	Dusseldorf	3	Tuesday	Annette	9.99	Jun	
	Dusseldorf	4	Wednesday	Manfred	86.81	Jul	
	London	6	Friday	Josie	74.02	Oct	
	Dusseldorf	1	Saturday	Manfred	43.11	Apr	

ANSWER:

```
USE shop;
```

```
UPDATE SALES1
SET SalesPerson = "Annette"
WHERE SalesPerson = "Inga" AND SalesAmount < 50;
```

3. Find out how many sales the London office logged
- Note we're looking for quantity here -
- (e.g. if London did 6 sales, then output would be 6)



ANSWER:
USE shop;

SELECT Store, COUNT(Store) as Sales
FROM SALES1
WHERE Store = "London";

4. Find the total (sum) sales amount by each person by day.

The screenshot shows a SQL IDE interface with a dark theme. The top bar indicates 'Local instance 3306'. The left sidebar shows a 'SCHEMAS' tree with 'shop' expanded, and 'SALES1' selected. The main editor displays a SQL query:

```
1 USE shop;  
2  
3 SELECT DISTINCT(SalesPerson), Day, CONCAT('£',Sum(SalesAmount)) as 'Total_Sales'  
4 FROM SALES1  
5 GROUP BY SalesPerson, Day;
```

Below the query editor, the 'Result Grid' shows the results of the query. The columns are 'SalesPerson', 'Day', and 'Total_Sales'. The results are as follows:

SalesPerson	Day	Total_Sales
Frank	Monday	£56.25
Frank	Tuesday	£74.32
Bill	Monday	£98.42
Bill	Saturday	£73.90
Josie	Tuesday	£44.27
Manfred	Monday	£77.00
Annette	Tuesday	£9.99
Manfred	Wednesday	£86.61
Josie	Friday	£74.02
Manfred	Saturday	£43.11

ANSWER:

```
USE shop;  
SELECT DISTINCT(SalesPerson), Day, CONCAT('£',Sum(SalesAmount)) as 'Total_Sales'  
FROM SALES1  
GROUP BY SalesPerson, Day;
```

5. How much (sum) each person sold for between week 1 and week 3.

The screenshot shows a database management interface with a dark theme. The top bar indicates 'Local instance 3306'. The left sidebar shows a tree view of schemas, with 'shop' expanded under 'Parts'. The main editor displays a SQL query:

```
1 USE shop;  
2  
3 SELECT Week, SalesPerson, SalesAmount As Sales  
4 FROM SALES1  
5 WHERE Week BETWEEN 1 AND 3  
6 ORDER By Week ASC;  
7  
8  
9
```

Below the query editor, the 'Result Grid' shows the results of the query. The columns are 'SalesPerson' and 'No_Of_Sales'. The results are as follows:

SalesPerson	No_Of_Sales
Frank	2
Bill	2
Josie	2
Annette	1

ANSWER:

USE shop;

```
SELECT Week, SalesPerson, SalesAmount As Sales  
FROM SALES1  
WHERE Week BETWEEN 1 AND 3  
ORDER By Week ASC;
```

6. For each store:

- The total of their sales;
- The number of sales;
- Their average sales;
- Their lowest sales amount;
- Their highest sales amount.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Schemas' pane is expanded, showing a tree view of databases including 'shop'. The 'shop' database is selected, and its 'Tables' folder is expanded, showing 'SALES1'. The 'Task2.6answer' query is selected in the 'Queries' pane. The query editor shows the following SQL code:

```
1 USE shop;
2
3 SELECT
4     Store,
5     SUM(SalesAmount) As Total_Number_Of_Sale,
6     COUNT(SalesAmount) As Number_Of_Sale,
7     CONCAT('£',ROUND((AVG(SalesAmount)),2)) AS Ave_Sales,
8     CONCAT('£',MIN(SalesAmount)) AS Min_Sales,
9     CONCAT('£',MAX(SalesAmount)) AS Max_Sales
10
11 FROM SALES1
12 GROUP by Store
13 ORDER by Store ASC;
14
15
```

Below the query editor, the 'Result Grid' is displayed, showing the results of the query. The grid has 7 columns: Store, Total_Number_Of_Sale, Number_Of_Sale, Ave_Sales, Min_Sales, and Max_Sales. The results are as follows:

Store	Total_Number_Of_Sale	Number_Of_Sale	Ave_Sales	Min_Sales	Max_Sales
Dusseldorf	216.91	4	£54.23	£9.99	£86.81
London	421.18	6	£70.20	£44.27	£98.42

USE shop;

```
SELECT
    Store,
    SUM(SalesAmount) As Total_Number_Of_Sale,
    COUNT(SalesAmount) As Number_Of_Sale,
    CONCAT('£',ROUND((AVG(SalesAmount)),2)) AS Ave_Sales,
    CONCAT('£',MIN(SalesAmount)) AS Min_Sales,
    CONCAT('£',MAX(SalesAmount)) AS Max_Sales
```

```
FROM SALES1
GROUP by Store
ORDER by Store ASC;
```


7. Find the average (AVG) monetary sales amount achieved by each store.

The screenshot shows the SQL Enterprise Manager interface. The left pane displays the 'SCHEMAS' tree with 'shop' expanded, showing 'SALES1'. The center pane contains the following SQL query:

```
1 USE shop;
2
3 SELECT
4     Store,
5     CONCAT('£',ROUND((AVG(SalesAmount)),2)) AS Ave_Sales
6 FROM SALES1
7 Group by Store;
```

The bottom pane shows the 'Result Grid' with the following data:

Store	Ave_Sales
London	£70.20
Dusseldorf	£54.23

ANSWER:

USE shop;

SELECT

Store,

CONCAT('£',ROUND((AVG(SalesAmount)),2)) AS Ave_Sales

FROM SALES1

Group by Store;

8. Count the number of sales by each person if they had less than 3 sales for the past period.

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Schemas' tree with 'shop' expanded, showing 'SALES1'. The center pane contains the following SQL query:

```
1 USE shop;  
2  
3 SELECT SalesPerson, COUNT(SalesAmount) As Number_Of_Sales  
4 FROM SALES1  
5 GROUP BY SalesPerson  
6 HAVING Number_Of_Sales < 3;  
7  
8
```

The bottom pane shows the 'Result Grid' with the following data:

SalesPerson	Number_Of_Sales
Frank	2
Bill	2
Josie	2
Annette	1

ANSWER:

USE shop;

```
SELECT SalesPerson, COUNT(SalesAmount) As Number_Of_Sales  
FROM SALES1  
GROUP BY SalesPerson  
HAVING Number_Of_Sales < 3;
```

9. Find the number (count) of sales by each person, but only if they made less than or equal to £300 worth of sales for the past period.

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Schemas' tree with 'shop' expanded, showing 'Tables' and 'SALES1'. The right pane shows a query window with the following SQL code:

```
1 USE shop;  
2  
3 SELECT SalesPerson,SUM(SalesAmount) As Total_Sales,COUNT(SalesAmount) As Number_Of_Sales  
4 FROM SALES1  
5 GROUP BY SalesPerson  
6 HAVING Total_Sales <= 300  
7 ORDER By SalesPerson ASC;  
8
```

Below the query window, the 'Result Grid' shows the results of the query. The columns are 'SalesPerson', 'Total_Sales', and 'Number_Of_Sales'. The results are as follows:

SalesPerson	Total_Sales	Number_Of_Sales
Annette	9.99	1
Bill	172.32	2
Frank	130.57	2
Josie	118.29	2
Manfred	206.92	3

ANSWER:
USE shop;

```
SELECT SalesPerson,SUM(SalesAmount) As Total_Sales,COUNT(SalesAmount) As Number_Of_Sales  
FROM SALES1  
GROUP BY SalesPerson  
HAVING Total_Sales <= 300  
ORDER By SalesPerson ASC;
```

TASK 3

USE PARTS DB TO WRITE THE FOLLOWING QUERIES

1. Return the PartID, Colour and Supplier name, where the supplier's surname ends in an S, and the Supplier city is not London. Ensure the values are Unique.

The screenshot shows a SQL Enterprise Manager interface for a 'Local instance 3306'. The 'Schemas' pane on the left shows a tree view with 'Parts' expanded, containing 'PART', 'PROJECT', 'SUPPLIER', and 'SUPPLY'. The main query editor displays the following SQL query:

```
1 USE Parts;
2
3 SELECT DISTINCT p.P_ID AS PARTID ,p.COLOUR AS 'Colour',s.SNAME AS 'Supplier Name'
4 FROM SUPPLIER AS s,part AS p
5 WHERE s.SNAME like '%S' and s.CITY <> 'LONDON'
6 ORDER BY p.P_ID ASC;
```

The 'Result Grid' at the bottom shows the query results in a table with columns: PARTID, Colour, and Supplier Name. The results are as follows:

PARTID	Colour	Supplier Name
P1	RED	ADAMS
P1	RED	JONES
P2	GREEN	ADAMS
P2	GREEN	JONES
P3	BLUE	ADAMS
P3	BLUE	JONES
P4	RED	ADAMS
P4	RED	JONES
P5	BLUE	ADAMS
P5	BLUE	JONES
P6	RED	ADAMS
P6	RED	JONES

ANSWER:

USE Parts;

```
SELECT DISTINCT p.P_ID AS PARTID ,p.COLOUR AS 'Colour',s.SNAME AS 'Supplier Name'
FROM SUPPLIER AS s,part AS p
WHERE s.SNAME like '%S' and s.CITY <> 'LONDON'
ORDER BY p.P_ID ASC;
```

2. Return the supplier name, part name and project name for each case where the following conditions are true:

i. The supplier supplies a project with a part;

The screenshot shows a database management tool interface. On the left, a tree view displays the database schema, including tables like Bakery, bank, customers, JOINS_PRACTICE, Parts, and shop. The 'PART' table is selected. The main area displays a SQL query:

```
1 USE Parts;
2
3 SELECT DISTINCT s.SNAME AS 'Supplier Name',p.PNAME AS 'Parts Name',pr.JNAME AS 'Project Name'
4 FROM supplier AS s
5 INNER JOIN supply AS su ON su.S_ID= s.S_ID
6 INNER JOIN part AS p ON p.P_ID = su.P_ID
7 INNER JOIN project AS pr ON pr.J_ID = su.J_ID
8 ORDER BY s.SNAME ASC;
```

Below the query, the 'Result Grid' shows the results of the query. The columns are 'Supplier Name', 'Parts Name', and 'Project Name'. The results are as follows:

Supplier Name	Parts Name	Project Name
ADAMS	COG	DISPLAY
ADAMS	BOLT	DISPLAY
ADAMS	COG	CONSOLE
ADAMS	CAM	CONSOLE
ADAMS	SCREW	CONSOLE
ADAMS	BOLT	CONSOLE
ADAMS	NUT	CONSOLE
ADAMS	CAM	RAID
ADAMS	CAM	TAPE
BLAKE	SCREW	SORTER
BLAKE	SCREW	DISPLAY
CLARK	COG	OCR
CLARK	COG	TAPE
JONES	SCREW	SORTER
JONES	CAM	DISPLAY
JONES	SCREW	DISPLAY
JONES	SCREW	OCR
JONES	SCREW	CONSOLE
JONES	SCREW	RAID
JONES	SCREW	EDS
JONES	SCREW	TAPE
SMITH	NUT	SORTER
SMITH	NUT	CONSOLE

ANSWER:

USE Parts;

SELECT DISTINCT s.SNAME AS 'Supplier Name',p.PNAME AS 'Parts Name',pr.JNAME AS 'Project Name'

FROM supplier AS s

INNER JOIN supply AS su ON su.S_ID= s.S_ID

INNER JOIN part AS p ON p.P_ID = su.P_ID

INNER JOIN project AS pr ON pr.J_ID = su.J_ID

ORDER BY s.SNAME ASC;

ii. And where the supplier's city, project city and part city are the same.

The screenshot shows a database management interface with a dark theme. On the left, a 'SCHEMAS' sidebar lists various database objects, with 'PART' selected under the 'Tables' category. The main window displays a SQL query in a text editor, which is a SELECT DISTINCT statement joining supplier, part, and project tables based on city. Below the query editor, a 'Result Grid' shows the output of the query as a table with 7 columns: Supplier Name, Parts Name, Project Name, Supplier City, Parts City, Project City, and an empty column. The results are sorted by Supplier Name in ascending order.

```
1 Use Parts;
2 SELECT DISTINCT
3     s.SNAME AS 'Supplier Name',
4     p.PNAME AS 'Parts Name',
5     pr.JNAME AS 'Project Name',
6     s.CITY AS 'Supplier City',
7     p.CITY AS 'Parts City',
8     pr.CITY AS 'Project City'
9 FROM
10     supplier AS s
11     INNER JOIN
12     part AS p ON p.CITY = s.CITY
13     INNER JOIN
14     project AS pr ON pr.CITY = s.CITY
15 ORDER BY s.SNAME ASC;
```

Supplier Name	Parts Name	Project Name	Supplier City	Parts City	Project City	
BLAKE	CAM	SORTER	PARIS	PARIS	PARIS	
BLAKE	BOLT	SORTER	PARIS	PARIS	PARIS	
CLARK	COG	RAID	LONDON	LONDON	LONDON	
CLARK	SCREW	RAID	LONDON	LONDON	LONDON	
CLARK	NUT	RAID	LONDON	LONDON	LONDON	
CLARK	COG	TAPE	LONDON	LONDON	LONDON	
CLARK	SCREW	TAPE	LONDON	LONDON	LONDON	
CLARK	NUT	TAPE	LONDON	LONDON	LONDON	
JONES	CAM	SORTER	PARIS	PARIS	PARIS	
JONES	BOLT	SORTER	PARIS	PARIS	PARIS	
SMITH	COG	RAID	LONDON	LONDON	LONDON	
SMITH	SCREW	RAID	LONDON	LONDON	LONDON	
SMITH	NUT	RAID	LONDON	LONDON	LONDON	
SMITH	COG	TAPE	LONDON	LONDON	LONDON	
SMITH	SCREW	TAPE	LONDON	LONDON	LONDON	
SMITH	NUT	TAPE	LONDON	LONDON	LONDON	

ANSWER:

Use Parts;

SELECT DISTINCT

s.SNAME AS 'Supplier Name',

p.PNAME AS 'Parts Name',

pr.JNAME AS 'Project Name',

s.CITY AS 'Supplier City',

p.CITY AS 'Parts City',

pr.CITY AS 'Project City'

FROM

supplier AS s

INNER JOIN

part AS p ON p.CITY = s.CITY

INNER JOIN

project AS pr ON pr.CITY = s.CITY

ORDER BY s.SNAME ASC;