Database Management Systems

Group Assignment 4

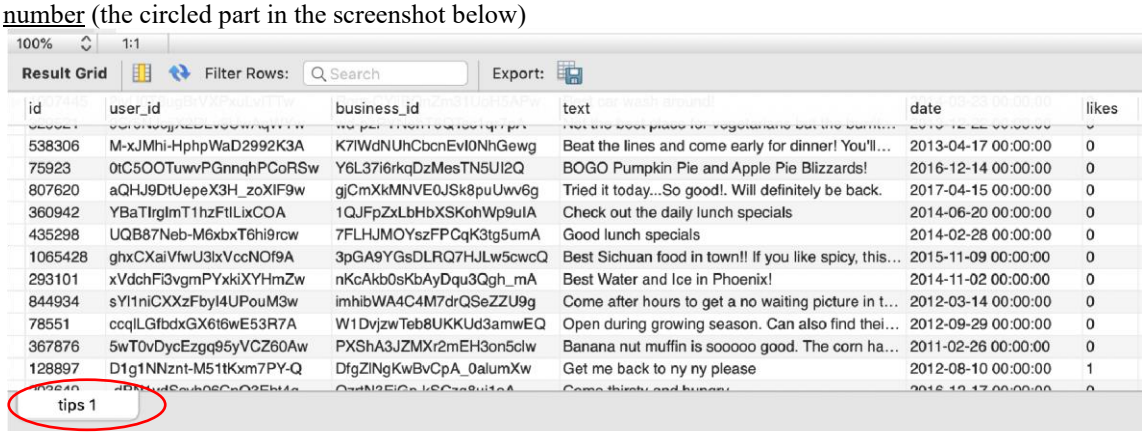
Deadline: Before the start of class, May 4th, 2025, at 11:59 PM Total Points: 40

Submission Instructions: Please include your group name, the members' names, and their email IDs. Your answers should be typed—formatted with a 1-inch margin, 12-point font, and single spacing. Add your answers beneath the appropriate question number. *Prepare a single pdf and name the file using your group name*. Each group should email (sparames@binghamton.edu) me the pdf before the deadline, copying all group members. Subject line: *Course Number – Assignment 4 – Spring 2025*

Important note for this assignment: Whenever you are asked to paste your "**output,"** you need to provide the following three items for each query:

(1) Paste the query used. Like this: SELECT * FROM yelp.tips;

(2) Paste the screenshot of the results table that MySQL displays when you execute your query – the results table should *clearly* show the output number (the circled part in the screenshot below)



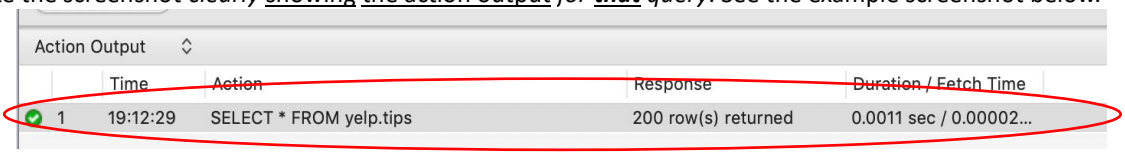
100% 1:1

Result Grid Filter Rows: Search Export:

id	user_id	business_id	text	date	likes
538306	M-xJMhi-HphpWaD2992K3A	K7lWdNUhCbcnEvI0NhGewg	Beat the lines and come early for dinner! You'll...	2013-04-17 00:00:00	0
75923	0tC5OOTuwvPGnnqhPCoRSw	Y6L37i6rkqDzMesTN5Ul2Q	BOGO Pumpkin Pie and Apple Pie Blizzards!	2016-12-14 00:00:00	0
807620	aQHJ9DtUepeX3H_zoXIF9w	gjCmXkMNVE0JSk8puUwv6g	Tried it today...So good!. Will definitely be back.	2017-04-15 00:00:00	0
360942	YBaTIrgImT1hzFtILixCOA	1QJFpZxLbHbXSKohWp9uIA	Check out the daily lunch specials	2014-06-20 00:00:00	0
435298	UQB87Neb-M6xbxT6hi9rcw	7FLHJMOYszFPCqK3tg5umA	Good lunch specials	2014-02-28 00:00:00	0
1065428	ghxCXaiVfwU3lxVccNOf9A	3pGA9YGsDLRQ7HJLw5cwcQ	Best Sichuan food in town!! If you like spicy, this...	2015-11-09 00:00:00	0
293101	xVdchFi3vgmPYxkiXYHmZw	nKcAkb0sKbAyDqu3Qgh_mA	Best Water and Ice in Phoenix!	2014-11-02 00:00:00	0
844934	sYl1niCXXzFbyI4UPouM3w	imhibWA4C4M7drQSeZZU9g	Come after hours to get a no waiting picture in t...	2012-03-14 00:00:00	0
78551	ccqlLGfbdxGX6t6wE53R7A	W1DvjzwTeb8UKKUd3amwEQ	Open during growing season. Can also find thei...	2012-09-29 00:00:00	0
367876	5wT0vDycEzgq95yVCZ60Aw	PXShA3JZMXr2mEH3on5clw	Banana nut muffin is sooooo good. The corn ha...	2011-02-26 00:00:00	0
128897	D1g1NNznt-M51tKxm7PY-Q	DfgZlNgKwBvCpA_0alumXw	Get me back to ny ny please	2012-08-10 00:00:00	1

tips 1

(3) paste the screenshot *clearly* showing the action output *for **that** query*. See the example screenshot below.



Action Output

	Time	Action	Response	Duration / Fetch Time
1	19:12:29	SELECT * FROM yelp.tips	200 row(s) returned	0.0011 sec / 0.00002...

Important Note: Please Read Carefully

To ensure academic integrity and promote independent learning, the following policy applies:

1. **No-AI Use Declaration:** By submitting this assignment, you confirm that absolutely NO AI tools (such as ChatGPT) were used in any part of your work. Any use of AI will be treated as a breach of academic integrity and will be subject to academic penalties.
2. **Preparedness for Oral Examination:** If AI use is suspected, you may be required to come to my office to explain your approach, demonstrate your understanding, and confirm the originality of your work.
3. **Detailed Comments Required:** All assignments must be thoroughly documented, with comments explaining your thought process and approach. Each line of code should have at least one line of explanation (in your own words) provided as comments besides, before, or after the line of code.
4. **Group Work Responsibility:** In group assignments, all members are responsible for the submitted work, including understanding and verifying every part. This responsibility applies regardless of individual contributions, and any team member should be able to answer questions on the entire assignment.
5. **Use of Class Material and Concepts:** Complete all assignments using only the material and concepts taught in class, ensuring that your work directly reflects your understanding of course content. If you use concepts not covered in class, explain in detail each technique used and why you chose it over class-taught concepts.
6. **Code Relevance and Consistency:** Ensure your code solutions directly address the assignment requirements, focusing on relevant elements that align with the task objectives. If you use non-standard techniques,

provide a clear explanation in your comments. Maintain consistency across assignment questions to demonstrate a coherent approach.

You are not supposed to collaborate with other groups on your assignments. This includes texting/emailing your solution, looking at other's solution for providing/seeking solution, and asking someone outside your group to debug your programs or solve the assignment. The assignment you submit should be your group's own work based on your group's study and/or research. You should acknowledge all material and sources used in preparation of your assignment, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. Plagiarizing other group's work or referring to the answers to these questions posted online (for ex: in crowdsourcing sites) or using a generative AI tool and/or submitting such work will not be tolerated. Such incidents will be subject to the maximum penalty applicable according to the Binghamton University guidelines for implementation of the code of academic honesty. By submitting your assignment, you also certify that your assignment has not previously been submitted for assessment in any other course, except where specific permission has been granted from all course coordinators involved, or at any other time in this course, and that you have not copied in part or whole or otherwise plagiarized the work of other students and/or persons outside your group. In addition, posting yours and/or the instructor's solution to assignments in the world wide web is not permitted.

Setup

We have been working on the *premiere* database in our class. We will work on this database for this assignment as well. You will use the script *assignment4_premiere.sql* (posted in the Group Assignments section on Brightspace) to create the *assignment4_premiere* database and its tables. Once you execute the *assignment4_premiere.sql* script, you are ready to answer the following questions.

Questions (each 1 point)

1. Use the CREATE TABLE command to create a table for the ORDER_LINE table, with the attributes displayed in Table 1 (see below). Remember that *order_num* and *part_num* together comprise this table's primary key. So, define a composite primary key. Ensure that (1) *order_num* is a foreign key referencing the *order_num* in ORDERS, and (2) *part_num* is a foreign key referencing the *part_num* in PART.

Table 1: Order Line table

ORDER_NUM	PART_NUM	NUM_ORDERED	QUOTED_PRICE
21608	AT94	11	21.95
21610	DR93	1	495.00
21610	DW11	1	399.99
21613	KL62	4	329.95
21614	KT03	2	595.00
21617	BV06	2	794.95
21617	CD52	4	150.00
21619	DR93	1	495.00
21623	KV29	2	1290.00

Copy the code you used to create the table and paste it here.

```
 CREATE TABLE ORDER_LINE
(ORDER_NUM CHAR(5),
PART_NUM CHAR(4),
NUM_ORDERED CHAR(3),
QUOTED_PRICE DECIMAL(6,2),
PRIMARY KEY (ORDER_NUM, PART_NUM),
CONSTRAINT FK_ORDER_NUM FOREIGN KEY (ORDER_NUM) REFERENCES
ORDERS(ORDER_NUM),
CONSTRAINT FK_PART_NUM FOREIGN KEY (PART_NUM) REFERENCES PART(PART_NUM)
);
DESC ORDER_LINE;
```

Field	Type	Null	Key	Default	Extra
ORDER_NUM	char(5)	NO	PRI	NULL	
PART_NUM	char(4)	NO	PRI	NULL	
NUM_ORDERED	char(3)	YES		NULL	
QUOTED_PRICE	decimal(6,2)	YES		NULL	

Result 8

45	22:00:41	CREATE TABLE ORDER_LINE (ORDER_NUM CH...	0 row(s) affected	0.031 sec
46	22:02:52	DESC ORDER_LINE	4 row(s) returned	0.000 sec / 0.000 sec

2. Use the INSERT INTO command to populate the table you created in question 1. Use the data from the table above to fill in the values. Copy and paste the code you used to populate the table.

```
INSERT INTO ORDER_LINE
VALUES ('21608', 'AT94', '11', 21.95);

INSERT INTO ORDER_LINE
VALUES ('21610', 'DR93', '1', 495.00);

INSERT INTO ORDER_LINE
VALUES ('21610', 'DW11', '1', 399.99);

INSERT INTO ORDER_LINE
VALUES ('21613', 'KL62', '4', 329.95);

INSERT INTO ORDER_LINE
VALUES ('21614', 'KT03', '2', 595.00);

INSERT INTO ORDER_LINE
VALUES ('21617', 'BV06', '2', 794.95);

INSERT INTO ORDER_LINE
VALUES ('21617', 'CD52', '4', 150.00);

INSERT INTO ORDER_LINE
VALUES ('21619', 'DR93', '1', 495.00);

INSERT INTO ORDER_LINE
VALUES ('21623', 'KV29', '2', 1290.00);
```

47	22:10:03	INSERT INTO ORDER_LINE VALUES ('21608', 'AT...	1 row(s) affected	0.016 sec
48	22:10:03	INSERT INTO ORDER_LINE VALUES ('21610', 'DR...	1 row(s) affected	0.000 sec
49	22:10:03	INSERT INTO ORDER_LINE VALUES ('21610', 'D...	1 row(s) affected	0.000 sec
50	22:10:03	INSERT INTO ORDER_LINE VALUES ('21613', 'KL...	1 row(s) affected	0.000 sec
51	22:10:03	INSERT INTO ORDER_LINE VALUES ('21614', 'KT...	1 row(s) affected	0.000 sec
52	22:10:03	INSERT INTO ORDER_LINE VALUES ('21617', 'BV...	1 row(s) affected	0.000 sec
53	22:10:04	INSERT INTO ORDER_LINE VALUES ('21617', 'CD...	1 row(s) affected	0.000 sec
54	22:10:04	INSERT INTO ORDER_LINE VALUES ('21619', 'DR...	1 row(s) affected	0.000 sec
55	22:10:04	INSERT INTO ORDER_LINE VALUES ('21623', 'KV...	1 row(s) affected	0.000 sec

3. Use the SELECT * command to display the table you created with the data values in them. Copy and paste your code.

SELECT * FROM ORDER_LINE;

ORDER_NUM	PART_NUM	NUM_ORDERED	QUOTED_PRICE
21608	AT94	11	21.95
21610	DR93	1	495.00
21610	DW11	1	399.99
21613	KL62	4	329.95
21614	KT03	2	595.00
21617	BV06	2	794.95
21617	CD52	4	150.00
21619	DR93	1	495.00
21623	KV29	2	1290.00
NULL	NULL	NULL	NULL

ORDER_LINE 46

56	22:11:09	SELECT * FROM ORDER_LINE LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec

For the questions below, copy and paste your output (see instructions on page 1).

4. Using a MySQL query, list all the rows for the low-availability parts. Low availability: If the number of parts on hand is less than or equal to 25, then we consider there is low availability.

SELECT * FROM PART WHERE ON_HAND <= 25;

PART_NUM	DESCRIPTION	ON_HAND	CLASS	WAREHOUSE	PRICE	ASSET_VALUE	CODE	NEW_CODE
DL71	Cordless Drill	21	HW	3	129.95	2728.95	NULL	DLHW
DR93	Gas Range	8	AP	2	495.00	3960.00	NULL	DRAP
DW11	Washer	12	AP	3	399.99	4799.88	NULL	DWAP
FD21	Stand Mixer	22	HW	3	159.95	3518.90	NULL	FDHW
KL62	Dryer	12	AP	1	349.95	4199.40	NULL	KLAP
KT03	Dishwasher	8	AP	3	595.00	4760.00	NULL	KTAP
KV29	Treadmill	9	SG	2	1390.00	12510.00	NULL	KVSG
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

PART 47

57 22:15:19 SELECT * FROM PART WHERE ON_HAND <= 25 ... 7 row(s) returned 0.016 sec / 0.000 sec

5. Using a MySQL query, list the description of all the parts with either the price greater than $400 or the description contains two words.

SELECT DESCRIPTION FROM PART WHERE PRICE > 400 OR DESCRIPTION LIKE "% %";

DESCRIPTION
Home Gym
Microwave Oven
Cordless Drill
Gas Range
Stand Mixer
Dishwasher
Treadmill

PART 48

58 22:24:52 SELECT DESCRIPTION FROM PART WHERE PRI... 7 row(s) returned 0.000 sec / 0.000 sec

6. Using a MySQL query, list the descriptions of the parts in the warehouses other than 4. Write three queries that can answer this question using the following requirements:
    a. Query 1 should use <>

SELECT DESCRIPTION FROM PART WHERE WAREHOUSE <> 4;

	DESCRIPTION
▶	Iron
	Home Gym
	Microwave Oven
	Cordless Drill
	Gas Range
	Washer
	Stand Mixer
	Dryer
	Dishwasher
	Treadmill

PART 49 × PART 50 PART 51 PART 52

93 12:03:29 SELECT DESCRIPTION FROM PART WHERE ... 10 row(s) returned 0.000 sec / 0.000 sec

b. Query 2 should use !=

SELECT DESCRIPTION FROM PART WHERE WAREHOUSE != 4;

	DESCRIPTION
▶	Iron
	Home Gym
	Microwave Oven
	Cordless Drill
	Gas Range
	Washer
	Stand Mixer
	Dryer
	Dishwasher
	Treadmill

PART 49 PART 50 × PART 51 PART 52

60 22:30:47 SELECT DESCRIPTION FROM PART WHERE WA... 10 row(s) returned 0.000 sec / 0.000 sec

c. Query 3 should use NOT

SELECT DESCRIPTION FROM PART WHERE WAREHOUSE NOT IN (4);

SELECT DESCRIPTION FROM PART WHERE WAREHOUSE NOT LIKE 4;

	DESCRIPTION
▶	Iron
	Home Gym
	Microwave Oven
	Cordless Drill
	Gas Range
	Washer
	Stand Mixer
	Dryer
	Dishwasher
	Treadmill

PART 49 PART 50 PART 51 × PART 52

	DESCRIPTION
▶	Iron
	Home Gym
	Microwave Oven
	Cordless Drill
	Gas Range
	Washer
	Stand Mixer
	Dryer
	Dishwasher
	Treadmill

PART 49 PART 50 PART 51 PART 52 ×

62	22:31:43	SELECT DESCRIPTION FROM PART WHERE WA...	10 row(s) returned	0.000 sec / 0.000 sec
63	22:32:08	SELECT DESCRIPTION FROM PART WHERE WA...	10 row(s) returned	0.000 sec / 0.000 sec

7. Using a MySQL query, find the part numbers, descriptions, and prices of all parts stored in warehouse '2', that have an on-hand quantity greater than 20, and are priced below $150.00.

SELECT PART_NUM, DESCRIPTION, PRICE FROM PART WHERE WAREHOUSE = "2" AND ON_HAND > 20 AND PRICE < 150.00;

	PART_NUM	DESCRIPTION	PRICE
*	NULL	NULL	NULL

PART 53

65 22:37:02 SELECT PART_NUM, DESCRIPTION, PRICE FRO... 0 row(s) returned 0.000 sec / 0.000 sec

8. Using a MySQL query, list the part number, description, and asset value for each part whose asset value is at least $10,500. Use *on_hand*price* to calculate the asset value for each part and label the calculated column as *asset_value*.

SET SQL_SAFE_UPDATES = 0;
ALTER TABLE PART ADD COLUMN ASSET_VALUE DECIMAL(8,2);
UPDATE PART SET ASSET_VALUE = (PRICE * ON_HAND);
SELECT PART_NUM, DESCRIPTION, ASSET_VALUE FROM PART WHERE ASSET_VALUE > 10500.00;

PART_NUM	DESCRIPTION	ASSET_VALUE
BV06	Home Gym	35772.75
KV29	Treadmill	12510.00
NULL	NULL	NULL

PART 54

66	22:48:22	SET SQL_SAFE_UPDATES = 0	0 row(s) affected	0.015 sec
67	22:48:24	ALTER TABLE PART ADD COLUMN ASSET_VALU...	0 row(s) affected Records: 0 Duplicates: 0 Warnings...	0.031 sec
71	22:59:13	UPDATE PART SET ASSET_VALUE = (PRICE * ON...	10 row(s) affected Rows matched: 10 Changed: 10 ...	0.000 sec
73	23:00:33	SELECT PART_NUM, DESCRIPTION, ASSET_VAL...	2 row(s) returned	0.000 sec / 0.000 sec

9. The company managing the *premiere* database wanted to display a new code that combines the first two characters of the *part_num* column and the entire value of the *class* column. Write a MySQL query to display the *part_num* and *class* columns and the column with the new code. Sample output of the report is shown below. Requirements: You cannot use the CONCAT or CONCT_WS functions.

part_num	class	New_Code
AT94	HW	ATHW
BV06	SG	BVSG
CD52	AP	CDAP
DL71	HW	DLHW
DR93	AP	DRAP
DW11	AP	DWAP
FD21	HW	FDHW
KL62	AP	KLAP
KT03	AP	KTAP
KV29	SG	KVSG

```
ALTER TABLE PART ADD COLUMN NEW_CODE CHAR(4);
UPDATE PART SET NEW_CODE = INSERT(CLASS, 1, 0, SUBSTRING(PART_NUM, 1, 2));
SELECT PART_NUM, CLASS, NEW_CODE FROM PART;
```

PART_NUM	CLASS	NEW_CODE
AT94	HW	ATHW
BV06	SG	BVSG
CD52	AP	CDAP
DL71	HW	DLHW
DR93	AP	DRAP
DW11	AP	DWAP
FD21	HW	FDHW
KL62	AP	KLAP
KT03	AP	KTAP
KV29	SG	KVSG
NULL	NULL	NULL

PART 56

S

76	10:12:22	ALTER TABLE PART ADD COLUMN NEW_CODE ...	0 row(s) affected Records: 0 Duplicates: 0 Warnings...	0.046 sec
108	12:20:59	UPDATE PART SET NEW_CODE = INSERT(CL...	0 row(s) affected Rows matched: 10 Changed: 0 ...	0.000 sec
109	12:21:04	SELECT PART_NUM, CLASS, NEW_CODE FR...	10 row(s) returned	0.000 sec / 0.000 sec

10. Using a MySQL query, display all the rows and columns of the orders table. The display should be sorted by order date (descending order) and order number. Order date is the primary sort key.

```
SELECT * FROM ORDERS
ORDER BY ORDER_DATE DESC, ORDER_NUM;
```

ORDER_NUM	ORDER_DATE	CUSTOMER_NUM
21617	2007-10-23	608
21619	2007-10-23	148
21623	2007-10-23	608
21613	2007-10-21	408
21614	2007-10-21	282
21608	2007-10-20	148
21610	2007-10-20	356
NULL	NULL	NULL

ORDERS 57

110 12:24:01 SELECT * FROM ORDERS ORDER BY ORDER... 7 row(s) returned 0.016 sec / 0.000 sec

11. Using a MySQL query, display the (a) maximum of the asset value (asset value= price * on_hand), (b) minimum of the asset value, and the (c) count of all the parts in Warehouse 2 and with low availability. Low availability: If the number of parts on hand is less than or equal to 25, then we consider there is low availability.

```
SELECT MAX(ASSET_VALUE), MIN(ASSET_VALUE), COUNT(*)
FROM PART
WHERE WAREHOUSE = '2' AND ON_HAND <= 25;
```

	MAX(ASSET_VALUE)	MIN(ASSET_VALUE)	COUNT(*)
▶	12510.00	3960.00	2

Result 59 ×

111 12:30:21 SELECT MAX(ASSET_VALUE), MIN(ASSET_VALU... 1 row(s) returned 0.016 sec / 0.000 sec

12. Using a MySQL query, list the part number, description, and price of all the parts classified as AP and whose availability (on_hand) is lesser than or equal to average availability.

```
SELECT PART_NUM, DESCRIPTION, PRICE
FROM PART
WHERE CLASS = 'AP' AND ON_HAND <= (SELECT AVG(ON_HAND) FROM PART);
```

PART_NUM	DESCRIPTION	PRICE
DR93	Gas Range	495.00
DW11	Washer	399.99
KL62	Dryer	349.95
KT03	Dishwasher	595.00
NULL	NULL	NULL

PART 60

115 12:39:52 SELECT PART_NUM, DESCRIPTION, PRICE FRO... 4 row(s) returned 0.016 sec / 0.000 sec

13. Using a MySQL query, display how many different i.e. distinct classes the PART table has.
SELECT COUNT(DISTINCT(CLASS)) FROM PART;

COUNT(DISTINCT(CLASS))
3

Result 61

116 12:42:38 SELECT COUNT(DISTINCT(CLASS)) FROM PART ... 1 row(s) returned 0.015 sec / 0.000 sec

14. Using a MySQL query, display the unique class types for parts with a price greater than $10 and stored in the warehouse '2'.

```
SELECT COUNT(DISTINCT(CLASS))
FROM PART
WHERE PRICE > 10 AND WAREHOUSE = '2';
```

	COUNT(DISTINCT(CLASS))
▶	2

Result 62

117 12:44:06 SELECT COUNT(DISTINCT(CLASS)) FROM PART ... 1 row(s) returned 0.016 sec / 0.000 sec

15. Using a MySQL query, display part numbers and the total number of unique orders for each part number.

```
SELECT PART_NUM, COUNT(DISTINCT(ORDER_NUM))
FROM ORDER_LINE
GROUP BY PART_NUM;
```

PART_NUM	COUNT(DISTINCT(ORDER_NUM))
AT94	1
BV06	1
CD52	1
DR93	2
DW11	1
KL62	1
KT03	1
KV29	1

Result 63

123 16:21:22 SELECT PART_NUM, COUNT(DISTINCT(ORDER_... 8 row(s) returned 0.000 sec / 0.000 sec

16. Using a MySQL query, display part numbers and the total number of unique orders for each part number, sorted by the total number of orders in descending order.

```
SELECT PART_NUM, COUNT(DISTINCT(ORDER_NUM)) AS ORDER_NUM_QUANT
FROM ORDER_LINE
GROUP BY PART_NUM
ORDER BY ORDER_NUM_QUANT DESC;
```

PART_NUM	ORDER_NUM_QUANT
DR93	2
AT94	1
BV06	1
CD52	1
DW11	1
KL62	1
KT03	1
KV29	1

Result 64

124 16:23:01 SELECT PART_NUM, COUNT(DISTINCT(ORDER_... 8 row(s) returned 0.016 sec / 0.000 sec

17. Using a MySQL query, display part numbers and the total number of unique orders for each part number. Restrict your answers so that you show only those parts included in more than 1 unique order.

```
SELECT PART_NUM, COUNT(DISTINCT(ORDER_NUM)) AS ORDER_NUM_QUANT
FROM ORDER_LINE
GROUP BY PART_NUM
HAVING ORDER_NUM_QUANT > 1
ORDER BY ORDER_NUM_QUANT DESC;
```

	PART_NUM	ORDER_NUM_QUANT
▶	DR93	2

Result 65 ×

125 16:24:13 SELECT PART_NUM, COUNT(DISTINCT(ORDER_... 1 row(s) returned 0.000 sec / 0.000 sec

18. Using a MySQL query, display each part number and the total number of parts ordered for each part number, only including line items with a quoted price greater than $300. Sort the display by the total number of parts ordered in descending order.

```
SELECT PART_NUM, COUNT(DISTINCT(ORDER_NUM)) AS ORDER_NUM_QUANT,
    SUM(NUM_ORDERED) AS TOTAL_ORDERED
FROM ORDER_LINE
WHERE QUOTED_PRICE > 300
GROUP BY PART_NUM
ORDER BY TOTAL_ORDERED DESC;
```

	PART_NUM	ORDER_NUM_QUANT	TOTAL_ORDERED
▶	KL62	1	4
	BV06	1	2
	DR93	2	2
	KT03	1	2
	KV29	1	2
	DW11	1	1

Result 67 ×

130 16:33:36 SELECT PART_NUM, COUNT(DISTINCT(ORDER_... 6 row(s) returned 0.000 sec / 0.000 sec

19. Using a MySQL query, display each part number and the total number of parts ordered for each part number, only including line items where the quoted price is greater than $300 and parts with total number of parts ordered more than 2.

```
SELECT PART_NUM, COUNT(DISTINCT(ORDER_NUM)) AS ORDER_NUM_QUANT,
SUM(NUM_ORDERED) AS TOTAL_ORDERED
FROM ORDER_LINE
WHERE QUOTED_PRICE > 300
GROUP BY PART_NUM
HAVING TOTAL_ORDERED > 2
ORDER BY TOTAL_ORDERED DESC;
```

	PART_NUM	ORDER_NUM_QUANT	TOTAL_ORDERED
▶	KL62	1	4

Result 68 ×

131	16:36:45	SELECT PART_NUM, COUNT(DISTINCT(ORDER_...	1 row(s) returned	0.000 sec / 0.000 sec

20. Using a MySQL query, add a new column to the parts table. The table should contain the prices of each part, with a dollar ($) symbol before the price. Sample output of the table with the new column shown below:

PART_NUM	DESCRIPTION	ON_HAND	CLASS	WAREHOUSE	PRICE	Dollar_Price
AT94	Iron	50	HW	3	24.95	$24.95
BV06	Home Gym	45	SG	2	794.95	$794.95
CD52	Microwave Oven	32	AP	1	165.00	$165.00
DL71	Cordless Drill	21	HW	3	129.95	$129.95
DR93	Gas Range	8	AP	2	495.00	$495.00
DW11	Washer	12	AP	3	399.99	$399.99
FD21	Stand Mixer	22	HW	3	159.95	$159.95
KL62	Dryer	12	AP	1	349.95	$349.95
KT03	Dishwasher	8	AP	3	595.00	$595.00
KV29	Treadmill	9	SG	2	1390.00	$1390.00

```
ALTER TABLE PART ADD COLUMN Dollar_Price VARCHAR(10);
UPDATE PART SET Dollar_Price = CONCAT("$", PRICE);
SELECT * FROM PART;
```

PART_NUM	DESCRIPTION	ON_HAND	CLASS	WAREHOUSE	PRICE	ASSET_VALUE	CODE	NEW_CODE	Dollar_Price
AT94	Iron	50	HW	3	24.95	1247.50	NULL	ATHW	$24.95
BV06	Home Gym	45	SG	2	794.95	35772.75	NULL	BVSG	$794.95
CD52	Microwave Oven	32	AP	1	165.00	5280.00	NULL	CDAP	$165.00
DL71	Cordless Drill	21	HW	3	129.95	2728.95	NULL	DLHW	$129.95
DR93	Gas Range	8	AP	2	495.00	3960.00	NULL	DRAP	$495.00
DW11	Washer	12	AP	3	399.99	4799.88	NULL	DWAP	$399.99
FD21	Stand Mixer	22	HW	3	159.95	3518.90	NULL	FDHW	$159.95
KL62	Dryer	12	AP	1	349.95	4199.40	NULL	KLAP	$349.95
KT03	Dishwasher	8	AP	3	595.00	4760.00	NULL	KTAP	$595.00
KV29	Treadmill	9	SG	2	1390.00	12510.00	NULL	KVSG	$1390.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

PART 70 × Apply

155	16:59:45	ALTER TABLE PART ADD COLUMN Dollar_Price V...	0 row(s) affected Records: 0 Duplicates: 0 Warning...	0.047 sec
157	17:00:47	UPDATE PART SET Dollar_Price = CONCAT("$", P...	10 row(s) affected Rows matched: 10 Changed: 10 ...	0.016 sec
158	17:00:49	SELECT * FROM PART LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

21. Add a foreign key constraint without deleting the customer table so that the *rep_num* in the customer table links to the *rep_num* in the rep table. Set the delete rule for the update operation to null. Set the delete rule for the delete operation to cascade.

```
ALTER TABLE CUSTOMER
ADD CONSTRAINT FK_REP_NUM FOREIGN KEY (REP_NUM) REFERENCES REP(REP_NUM)
ON UPDATE SET NULL ON DELETE CASCADE;
```

5	17:19:31	ALTER TABLE CUSTOMER ADD CONSTRAINT ...	10 row(s) affected Records: 10 Duplicates: 0 War...	0.063 sec

22. Using a MySQL query, list all pairs of customers from the same city and state. Ensure that you list the cities and states of customers in each pair.

```
SELECT customer_name, CONCAT(first_name, " ", last_name) AS rep_name,
    CONCAT(CUSTOMER.city, ", ", CUSTOMER.state) AS location
FROM CUSTOMER, REP
```

WHERE CUSTOMER.rep_num = REP.rep_num AND CUSTOMER.city = REP.city AND
CUSTOMER.state = REP.state;

customer_name	rep_name	location
All Season	Valerie Kaiser	Grove, FL
Deerfield's Four Seasons	Richard Hull	Sheldon, FL

Result 10

11 14:09:33 SELECT customer_name, CONCAT(first_name, " ... 2 row(s) returned 0.000 sec / 0.000 sec

23. Using a MySQL query, list all the customer names and their corresponding representative's first and last names.

SELECT customer_name, CONCAT(first_name, " ", last_name) AS rep_name
FROM CUSTOMER, REP
WHERE CUSTOMER.rep_num = REP.rep_num;

customer_name	rep_name
Al's Appliance and Sport	Valerie Kaiser
Kline's	Valerie Kaiser
All Season	Valerie Kaiser
Brookings Direct	Richard Hull
The Everything Shop	Richard Hull
Lee's Sport and Appliance	Richard Hull
Deerfield's Four Seasons	Richard Hull
Ferguson's	Juan Perez
Bargains Galore	Juan Perez
Johnson's Department Store	Juan Perez

Result 12

14 14:13:16 SELECT customer_name, CONCAT(first_name, " ... 10 row(s) returned 0.000 sec / 0.000 sec

24. Using a MySQL query, list all orders (all columns) along with their customer names and cities for customers located in 'Fillmore'.

SELECT customer_name, city, ORDERS.* FROM customer, orders
WHERE CUSTOMER.customer_num = ORDERS.customer_num AND CUSTOMER.city = 'Fillmore';

customer_name	city	ORDER_NUM	ORDER_DATE	CUSTOMER_NUM
Al's Appliance and Sport	Fillmore	21608	2007-10-20	148
Al's Appliance and Sport	Fillmore	21619	2007-10-23	148

Result 14

16 14:26:31 SELECT customer_name, city, ORDERS.* FROM... 2 row(s) returned 0.000 sec / 0.000 sec

25. Using a MySQL query, list all parts (all columns) and their corresponding order line items (all columns) where the part's *class* is 'AP' and the quoted price is greater than $100.

SELECT * FROM part, order_line
WHERE PART.part_num = ORDER_LINE.part_num AND PART.class = 'AP' AND quoted_price > 100;

PART_NUM	DESCRIPTION	ON_HAND	CLASS	WAREHOUSE	PRICE	ASSET_VALUE	CODE	NEW_CODE	Dollar_Price	ORDER_NUM	PART_NUM	NUM_ORDERED	QUOTED_PRICE
CD52	Microwave Oven	32	AP	1	165.00	5280.00	NULL	CDAP	$165.00	21617	CD52	4	150.00
DR93	Gas Range	8	AP	2	495.00	3960.00	NULL	DRAP	$495.00	21610	DR93	1	495.00
DR93	Gas Range	8	AP	2	495.00	3960.00	NULL	DRAP	$495.00	21619	DR93	1	495.00
DW11	Washer	12	AP	3	399.99	4799.88	NULL	DWAP	$399.99	21610	DW11	1	399.99
KL62	Dryer	12	AP	1	349.95	4199.40	NULL	KLAP	$349.95	21613	KL62	4	329.95
KT03	Dishwasher	8	AP	3	595.00	4760.00	NULL	KTAP	$595.00	21614	KT03	2	595.00

Result 15

17 14:31:47 SELECT * FROM part, order_line WHERE PART.part_num = ORDER_LINE.part_num AND PART.class = 'AP' AND quoted_pric... 6 row(s) returned 0.016 sec / 0.000 sec

26. Using a MySQL query, list each representative's first and last names and the number of customers they represent.

SELECT first_name, last_name, COUNT(customer_num) FROM customer, rep
WHERE CUSTOMER.rep_num = REP.rep_num
GROUP BY REP.rep_num;

first_name	last_name	COUNT(customer_num)
Valerie	Kaiser	3
Richard	Hull	4
Juan	Perez	3

Result 19

22 14:40:27 SELECT first_name, last_name, COUNT(customer... 3 row(s) returned 0.000 sec / 0.000 sec

27. Using a MySQL query, list all parts (their *part_num* and *description* columns) that have been ordered more than once i.e., they are part of more than one unique order.

SELECT DISTINCT(PART.part_num), description FROM part, order_line
WHERE PART.part_num = ORDER_LINE.part_num AND PART.part_num IN (SELECT part_num FROM order_line GROUP BY PART_NUM HAVING COUNT(*) > 1);

part_num	description
DR93	Gas Range

Result 17

30 15:09:44 SELECT DISTINCT(PART.part_num), description ... 1 row(s) returned 0.000 sec / 0.000 sec

28. Using a MySQL query, list each *part_num* and the total quantity ordered for that *part_num* across all orders. Restrict your output to parts that cost more than $200.

SELECT PART.part_num, SUM(num_ordered) FROM part, order_line
WHERE PART.part_num = ORDER_LINE.part_num AND PART.price > 200.00
GROUP BY PART.part_num;

part_num	SUM(num_ordered)
BV06	2
DR93	2
DW11	1
KL62	4
KT03	2
KV29	2

Result 34

44 15:18:27 SELECT PART.part_num, SUM(num_ordered) FR... 6 row(s) returned 0.000 sec / 0.000 sec

29. Using a MySQL query, list each *part_num* and the total quantity ordered for that *part_num* across all orders. Restrict your output to parts costing more than $200 and quoted prices more than $50.

SELECT PART.part_num, SUM(num_ordered) FROM part, order_line
WHERE PART.part_num = ORDER_LINE.part_num AND PART.price > 200.00 AND quoted_price > 300.00
GROUP BY PART.part_num;

part_num	SUM(num_ordered)
DR93	2
DW11	1
KL62	4
KT03	2
BV06	2
KV29	2

Result 40

54 15:21:52 SELECT PART.part_num, SUM(num_ordered) FR... 6 row(s) returned 0.000 sec / 0.000 sec

30. Using a MySQL query, list each *part_num* and the total quantity ordered for that *part_num* across all orders. Restrict your output to (a) parts that cost more than $200 and quoted price of more than $50, and (b) parts with a total quantity ordered greater than 2.

```
SELECT PART.part_num, SUM(num_ordered) FROM part, order_line
WHERE PART.part_num = ORDER_LINE.part_num AND PART.price > 200.00 AND quoted_price >
50.00
GROUP BY PART.part_num
HAVING SUM(num_ordered) > 2;
```

	part_num	SUM(num_ordered)
▶	KL62	4

Result 43 ×

57 15:23:47 SELECT PART.part_num, SUM(num_ordered) FR... 1 row(s) returned 0.000 sec / 0.000 sec

31. Using a MySQL query, list the customers' names with a quoted price of at least $400.

```
SELECT customer_name, quoted_price FROM customer, orders, order_line
WHERE CUSTOMER.customer_num = ORDERS.customer_num AND ORDERS.order_num =
ORDER_LINE.order_num AND quoted_price > 400;
```

customer_name	quoted_price
Ferguson's	495.00
Brookings Direct	595.00
Johnson's Department Store	794.95
Al's Appliance and Sport	495.00
Johnson's Department Store	1290.00

Result 38

48 16:00:06 SELECT customer_name, quoted_price FROM cu... 5 row(s) returned 0.000 sec / 0.000 sec

32. Using a MySQL query, list the names of the customers who ordered on 2007-10-23. Requirement: Answer this question using joins.

SELECT customer_name, order_date FROM customer, orders
WHERE CUSTOMER.customer_num = ORDERS.customer_num AND order_date = '2007-10-23';

customer_name	order_date
Johnson's Department Store	2007-10-23
Al's Appliance and Sport	2007-10-23
Johnson's Department Store	2007-10-23

Result 40

50 16:01:56 SELECT customer_name, order_date FROM cust... 3 row(s) returned 0.000 sec / 0.000 sec

33. Using a MySQL query, list the names of the customers who ordered on 2007-10-23. Requirement: Answer this question using a subquery.

SELECT customer_name FROM customer
WHERE customer_num IN (SELECT customer_num FROM orders WHERE order_date = '2007-10-23');

customer_name
Johnson's Department Store
Al's Appliance and Sport

customer 43

53 16:05:24 SELECT customer_name FROM customer WHER... 2 row(s) returned 0.016 sec / 0.000 sec

34. Using a MySQL query, list all customers (all columns) along with any orders (all columns) they might have placed. Include customers who have not placed any orders.

SELECT C.*, O.* FROM customer C LEFT JOIN orders O
ON C.customer_num = O.customer_NUM;

CUSTOMER_NUM	CUSTOMER_NAME	STREET	CITY	STATE	ZIP	BALANCE	CREDIT_LIMIT	REP_NUM	ORDER_NUM	ORDER_DATE	CUSTOMER_NUM
282	Brookings Direct	3827 Devon	Grove	FL	33321	431.50	10000.00	35	21614	2007-10-21	282
356	Ferguson's	382 Wildwood	Northfield	FL	33146	5785.00	7500.00	65	21610	2007-10-20	356
408	The Everything Shop	1828 Raven	Crystal	FL	33503	5285.25	5000.00	35	21613	2007-10-21	408
462	Bargains Galore	3829 Central	Grove	FL	33321	3412.00	10000.00	65	NULL	NULL	NULL
524	Kline's	838 Ridgeland	Fillmore	FL	33336	12762.00	15000.00	20	NULL	NULL	NULL
608	Johnson's Department Store	372 Oxford	Sheldon	FL	33553	2106.00	10000.00	65	21623	2007-10-23	608
608	Johnson's Department Store	372 Oxford	Sheldon	FL	33553	2106.00	10000.00	65	21617	2007-10-23	608
687	Lee's Sport and Appliance	282 Evergreen	Altonville	FL	32543	2851.00	5000.00	35	NULL	NULL	NULL
725	Deerfield's Four Seasons	282 Columbia	Sheldon	FL	33553	248.00	7500.00	35	NULL	NULL	NULL
842	All Season	28 Lakeview	Grove	FL	33321	8221.00	7500.00	20	NULL	NULL	NULL

Result 4

6 17:48:54 SELECT C.*, O.* FROM customer C LEFT JOIN o... 12 row(s) returned 0.000 sec / 0.000 sec

35. Using a MySQL query, list all orders (all columns) along with the customer names, including orders that somehow have no associated customer records.

SELECT C.*, O.* FROM customer C RIGHT JOIN orders O
ON C.customer_num = O.customer_NUM;

CUSTOMER_NUM	CUSTOMER_NAME	STREET	CITY	STATE	ZIP	BALANCE	CREDIT_LIMIT	REP_NUM	ORDER_NUM	ORDER_DATE	CUSTOMER_NUM
148	Al's Appliance and Sport	2837 Greenway	Fillmore	FL	33336	6550.00	7500.00	20	21608	2007-10-20	148
356	Ferguson's	382 Wildwood	Northfield	FL	33146	5785.00	7500.00	65	21610	2007-10-20	356
408	The Everything Shop	1828 Raven	Crystal	FL	33503	5285.25	5000.00	35	21613	2007-10-21	408
282	Brookings Direct	3827 Devon	Grove	FL	33321	431.50	10000.00	35	21614	2007-10-21	282
608	Johnson's Department Store	372 Oxford	Sheldon	FL	33553	2106.00	10000.00	65	21617	2007-10-23	608
148	Al's Appliance and Sport	2837 Greenway	Fillmore	FL	33336	6550.00	7500.00	20	21619	2007-10-23	148
608	Johnson's Department Store	372 Oxford	Sheldon	FL	33553	2106.00	10000.00	65	21623	2007-10-23	608

Result 9

11 17:53:05 SELECT C.*, O.* FROM customer C RIGHT JOIN ... 7 row(s) returned 0.000 sec / 0.000 sec

36. Using a MySQL query, list all customers (customer names) and all orders (order numbers), showing the matching records where available and displaying NULLs where there is no match.

SELECT C.customer_name, O.order_num FROM customer C LEFT JOIN orders O
ON C.customer_num = O.customer_NUM
UNION
SELECT C.customer_name, O.order_num FROM customer C RIGHT JOIN orders O
ON C.customer_num = O.customer_NUM;

customer_name	order_num
Al's Appliance and Sport	21619
Al's Appliance and Sport	21608
Brookings Direct	21614
Ferguson's	21610
The Everything Shop	21613
Bargains Galore	NULL
Kline's	NULL
Johnson's Department Store	21623
Johnson's Department Store	21617
Lee's Sport and Appliance	NULL
Deerfield's Four Seasons	NULL
All Season	NULL

Result 10

12 17:57:15 SELECT C.customer_name, O.order_num FROM ... 12 row(s) returned 0.015 sec / 0.000 sec

37. Using a MySQL query, create a list of cities that includes every city from the customer and representative tables, without duplicates.

SELECT C.city FROM customer C
UNION
SELECT R.city from rep R;

city
Fillmore
Grove
Northfield
Crystal
Sheldon
Altonville

Result 19

21 18:20:57 SELECT C.city FROM customer C UNION SELE... 6 row(s) returned 0.000 sec / 0.000 sec

38. Using a MySQL query, find cities that are common between customers and representatives.

```
SELECT DISTINCT(C.city) FROM customer C
WHERE C.city IN
(SELECT R.city from rep R);
```

city
Fillmore
Grove
Sheldon

Result 21

23 18:22:58 SELECT DISTINCT(C.city) FROM customer C W... 3 row(s) returned 0.000 sec / 0.000 sec

39. Using a MySQL query, list cities where there are customers but no representatives.

```
SELECT DISTINCT(C.city) FROM customer C
WHERE C.city NOT IN
(SELECT R.city from rep R);
```

city
Northfield
Crystal
Altonville

Result 22

24 18:25:51 SELECT DISTINCT(C.city) FROM customer C W... 3 row(s) returned 0.000 sec / 0.000 sec

40. Using a MySQL query, list cities where there are representatives but no customers.

```
SELECT DISTINCT(R.city) FROM rep R
WHERE R.city NOT IN
(SELECT C.city from customer C);
```

	city

Result 23

25	18:26:51	SELECT DISTINCT(R.city) FROM rep R WHERE...	0 row(s) returned	0.000 sec / 0.000 sec