Exercise #1:

Based on the *employees* table below, select all fields from the *employees* table whose salary is less than or equal to $52,500 (no sorting is required):

```
CREATE TABLE employees
( employee_number int NOT NULL,
  last_name char(50) NOT NULL,
  first_name char(50) NOT NULL,
  salary int,
  dept_id int,
  CONSTRAINT employees_pk PRIMARY KEY (employee_number)
);

INSERT INTO employees
(employee_number, last_name, first_name, salary, dept_id)
VALUES
(1001, 'Smith', 'John', 62000, 500);
INSERT INTO employees
(employee_number, last_name, first_name, salary, dept_id)
VALUES
(1002, 'Anderson', 'Jane', 57500, 500);
INSERT INTO employees
(employee_number, last_name, first_name, salary, dept_id)
VALUES
(1003, 'Everest', 'Brad', 71000, 501);
INSERT INTO employees
(employee_number, last_name, first_name, salary, dept_id)
VALUES
(1004, 'Horvath', 'Jack', 42000, 501);
```

Exercise #2:

Based on the *suppliers* table below, select the unique *city* values that reside in the *state* of California and order the results in descending order by *city*:

```
CREATE TABLE suppliers
( supplier_id int NOT NULL,
  supplier_name char(50) NOT NULL,
  city char(50),
  state char(25),
  CONSTRAINT suppliers_pk PRIMARY KEY (supplier_id)
);

INSERT INTO suppliers
(supplier_id, supplier_name, city, state)
VALUES
(100, 'Microsoft', 'Redmond', 'Washington');

INSERT INTO suppliers
(supplier_id, supplier_name, city, state)
VALUES
(200, 'Google', 'Mountain View', 'California');

INSERT INTO suppliers
(supplier_id, supplier_name, city, state)
VALUES
(300, 'Oracle', 'Redwood City', 'California');

INSERT INTO suppliers
(supplier_id, supplier_name, city, state)
VALUES
(400, 'Kimberly-Clark', 'Irving', 'Texas');

INSERT INTO suppliers
(supplier_id, supplier_name, city, state)
VALUES
(500, 'Tyson Foods', 'Springdale', 'Arkansas');
```

```
INSERT INTO suppliers
(supplier_id, supplier_name, city, state)
VALUES
(600, 'SC Johnson', 'Racine', 'Wisconsin');

INSERT INTO suppliers
(supplier_id, supplier_name, city, state)
VALUES
(700, 'Dole Food Company', 'Westlake Village', 'California');

INSERT INTO suppliers
(supplier_id, supplier_name, city, state)
VALUES
(800, 'Flowers Foods', 'Thomasville', 'Georgia');

INSERT INTO suppliers
(supplier_id, supplier_name, city, state)
VALUES
(900, 'Electronic Arts', 'Redwood City', 'California');
```

Exercise #3:

Based on the *customers* table and the *orders* table below, select the *customer_id* and *last_name* from the *customers* table and select the *order_date* from the *orders* table where there is a matching *customer_id* value in both the *customers* and *orders* tables. Order the results by *customer_id* in descending order.

```
CREATE TABLE customers
```

```
( customer_id int NOT NULL,
  last_name char(50) NOT NULL,
  first_name char(50) NOT NULL,
  favorite_website char(50),
  CONSTRAINT customers_pk PRIMARY KEY (customer_id)
);

CREATE TABLE orders
( order_id int NOT NULL,
  customer_id int,
  order_date date,
  CONSTRAINT orders_pk PRIMARY KEY (order_id)
);

INSERT INTO customers
(customer_id, last_name, first_name, favorite_website)
VALUES
(4000, 'Jackson', 'Joe', 'techonthenet.com');

INSERT INTO customers
(customer_id, last_name, first_name, favorite_website)
VALUES
(5000, 'Smith', 'Jane', 'digminecraft.com');

INSERT INTO customers
(customer_id, last_name, first_name, favorite_website)
VALUES
(6000, 'Ferguson', 'Samantha', 'bigactivities.com');

INSERT INTO customers
(customer_id, last_name, first_name, favorite_website)
VALUES
(7000, 'Reynolds', 'Allen', 'checkyourmath.com');

INSERT INTO customers
```

```
(customer_id, last_name, first_name, favorite_website)
VALUES
(8000, 'Anderson', 'Paige', NULL);

INSERT INTO customers
(customer_id, last_name, first_name, favorite_website)
VALUES
(9000, 'Johnson', 'Derek', 'techonthenet.com');

INSERT INTO orders
(order_id, customer_id, order_date)
VALUES
(1,7000,'2016/04/18');

INSERT INTO orders
(order_id, customer_id, order_date)
VALUES
(2,5000,'2016/04/18');

INSERT INTO orders
(order_id, customer_id, order_date)
VALUES
(3,8000,'2016/04/19');

INSERT INTO orders
(order_id, customer_id, order_date)
VALUES
(4,4000,'2016/04/20');

INSERT INTO orders
(order_id, customer_id, order_date)
VALUES
(5,null,'2016/05/01');
```

Exercise #4:

Based on the *customers* and *orders* table from Practice Exercise #3, select the *customer_id* and *last_name* from the *customers* table where there is a record in the *orders* table for that *customer_id*. Order the results in ascending order by *last_name* and then descending order by *customer_id*.

```
CREATE TABLE customers
( customer_id int NOT NULL,
  last_name char(50) NOT NULL,
  first_name char(50) NOT NULL,
  favorite_website char(50),
  CONSTRAINT customers_pk PRIMARY KEY (customer_id)
);

CREATE TABLE orders
( order_id int NOT NULL,
  customer_id int,
  order_date date,
  CONSTRAINT orders_pk PRIMARY KEY (order_id)
);
```

items_ordered

customerid	order_date	item	quantity	price
10330	30-Jun-1999	Pogo stick	1	28.00
10101	30-Jun-1999	Raft	1	58.00
10298	01-Jul-1999	Skateboard	1	33.00
10101	01-Jul-1999	Life Vest	4	125.00
10299	06-Jul-1999	Parachute	1	1250.00
10339	27-Jul-1999	Umbrella	1	4.50
10449	13-Aug-1999	Unicycle	1	180.79
10439	14-Aug-1999	Ski Poles	2	25.50
10101	18-Aug-1999	Rain Coat	1	18.30
10449	01-Sep-1999	Snow Shoes	1	45.00
10439	18-Sep-1999	Tent	1	88.00
10298	19-Sep-1999	Lantern	2	29.00
10410	28-Oct-1999	Sleeping Bag	1	89.22
10438	01-Nov-1999	Umbrella	1	6.75

10438	02-Nov-1999	Pillow	1	8.50
10298	01-Dec-1999	Helmet	1	22.00
10449	15-Dec-1999	Bicycle	1	380.50
10449	22-Dec-1999	Canoe	1	280.00
10101	30-Dec-1999	Hoola Hoop	3	14.75
10330	01-Jan-2000	Flashlight	4	28.00
10101	02-Jan-2000	Lantern	1	16.00
10299	18-Jan-2000	Inflatable Mattress	1	38.00
10438	18-Jan-2000	Tent	1	79.99
10413	19-Jan-2000	Lawnchair	4	32.00
10410	30-Jan-2000	Unicycle	1	192.50
10315	2-Feb-2000	Compass	1	8.00
10449	29-Feb-2000	Flashlight	1	4.50
10101	08-Mar-2000	Sleeping Bag	2	88.70
10298	18-Mar-2000	Pocket Knife	1	22.38
10449	19-Mar-2000	Canoe paddle	2	40.00
10298	01-Apr-2000	Ear Muffs	1	12.50
10330	19-Apr-2000	Shovel	1	16.75

customers

customerid	firstname	lastname	city	state
10101	John	Gray	Lynden	Washington
10298	Leroy	Brown	Pinetop	Arizona
10299	Elroy	Keller	Snoqualmie	Washington
10315	Lisa	Jones	Oshkosh	Wisconsin
10325	Ginger	Schultz	Pocatello	Idaho
10329	Kelly	Mendoza	Kailua	Hawaii
10330	Shawn	Dalton	Cannon Beach	Oregon
10338	Michael	Howell	Tillamook	Oregon
10339	Anthony	Sanchez	Winslow	Arizona
10408	Elroy	Cleaver	Globe	Arizona
10410	Mary Ann	Howell	Charleston	South Carolina
10413	Donald	Davids	Gila Bend	Arizona
10419	Linda	Sakahara	Nogales	Arizona
10429	Sarah	Graham	Greensboro	North Carolina
10438	Kevin	Smith	Durango	Colorado
10439	Conrad	Giles	Telluride	Colorado
10449	Isabela	Moore	Yuma	Arizona

Exercises

1. From the *items_ordered* table, select a list of all items purchased for customerid 10449. Display the customerid, item, and price for this customer.
2. Select all columns from the *items_ordered* table for whoever purchased a **Tent**.
3. Select the customerid, order_date, and item values from the items_ordered table for any items in the item column that start with the letter "S".
4. Select the distinct items in the items_ordered table. In other words, display a listing of each of the unique items from the items_ordered table.
5. Make up your own select statements and submit them.
6. Select the maximum price of any item ordered in the items_ordered table. Hint: Select the maximum price only.
7. Select the average price of all of the items ordered that were purchased in the month of Dec.
8. What are the total number of rows in the items_ordered table?
9. For all of the tents that were ordered in the items_ordered table, what is the price of the lowest tent? Hint: Your query should return the price only.
10. How many people are in each unique state in the customers table? Select the state and display the number of people in each. Hint: **count** is used to count rows in a column, **sum** works on numeric data only.
11. From the items_ordered table, select the item, maximum price, and minimum price for each specific item in the table. Hint: The items will need to be broken up into separate groups.
12. How many orders did each customer make? Use the items_ordered table. Select the customerid, number of orders they made, and the sum of their orders. Click the Group By answers link below if you have any problems.
13. How many people are in each unique state in the customers table that have more than one person in the state? Select the state and display the number of how many people are in each if it's greater than 1.
14. From the items_ordered table, select the item, maximum price, and minimum price for each specific item in the table. Only display the results if the maximum price for one of the items is greater than 190.00.
15. How many orders did each customer make? Use the items_ordered table. Select the customerid, number of orders they made, and the sum of their orders if they purchased more than 1 item.
16. Select the lastname, firstname, and city for all customers in the customers table. Display the results in Ascending Order based on the lastname.
17. Same thing as exercise #1, but display the results in Descending order.
18. Select the item and price for all of the items in the items_ordered table that the price is greater than 10.00. Display the results in Ascending order based on the price.
19. Select the customerid, order_date, and item from the items_ordered table for all items unless they are 'Snow Shoes' or if they are 'Ear Muffs'. Display the rows as long as they are not either of these two items.
20. Select the item and price of all items that start with the letters 'S', 'P', or 'F'.
21. Select the date, item, and price from the items_ordered table for all of the rows that have a price value ranging from 10.00 to 80.00.
22. Select the firstname, city, and state from the customers table for all of the rows where the state value is either: Arizona, Washington, Oklahoma, Colorado, or Hawaii.

23. Select the item and per unit price for each item in the items_ordered table. Hint: Divide the price by the quantity.
24. Write a query using a join to determine which items were ordered by each of the customers in the customers table. Select the customerid, firstname, lastname, order_date, item, and price for everything each customer purchased in the items_ordered table.
25. Repeat exercise #1, however display the results sorted by state in descending order.