

Name: David Norman

Section: \_\_\_\_\_

Exam #2- ISM6217  
Spring 2005 Version A

94  
100

1. State four rules-of-thumb for deciding what indexes to create. (4 points)

Index primary keys  
foreign keys  
commonly used in WHERE clauses  
columns commonly selected together can  
become a combination key

2. How do inner and outer joins differ? (3 points)

Inner joins return matching instances of primary and foreign keys. Repeated instances are not displayed. Outer joins will display the primary key and the foreign key whether the foreign key is null or not.

3. What is the basic tradeoff that must be considered when deciding what indexes to create? (4 points)

Creating indexes takes up space and makes UPDATE queries take longer. Indexes can often increase the speed of SELECT queries, though.

Name: David Norman Section: \_\_\_\_\_

4. List three inputs into the physical design process. (3 points)

Logical design  
Documentation  
DBMS characteristics  
Response time requirements  
Security, backup, recovery, integrity requirements

5. Contrast clustered and un-clustered indexes. (3 points)



clustered indexes index on non-key data to organize the index for faster scanning and other querying - unclustered has much slower scanning

6. Name three general tips for improving the performance of multi-table queries. (3 points)

- 1) put larger tables last in the FROM section
- 2) put ~~table~~ table joins before specific ~~WHERE~~ clauses
- 3) using table aliases can speed things up

First in WHERE clauses

The remaining questions refer to the attached logical data model. For each question, write the SQL code that will produce the requested information or answer the stated question.

7. How many employees live in one of the following cities, Seattle, London, Orlando? (5 points)

```
SELECT COUNT(*) FROM EMPLOYEES WHERE  
City IN ('Seattle', 'London', 'Orlando');
```

Name: David Norman

Section: \_\_\_\_\_

8. For each order taken by employees with the ID 1, 3 or 5, list the employee ID, first name, and last name, along with the order ID and date, and company name of the customer placing the order. Put the list in ascending order of employee last name, then first name, then descending order of order date. (10 points)

```
SELECT Employees.EmployeeID, Employees.FirstName,
Employees.LastName, Orders.OrderID, Orders.OrderDate,
Customers.CompanyName FROM Orders, Employees, Customers
WHERE Orders.EmployeeID = Employees.EmployeeID AND
Orders.CustomerID = Customers.CustomerID AND
Employees.EmployeeID IN (1, 3, 5) ORDER BY
Employees.LastName ASC, Employees.FirstName ASC,
Orders.OrderDate DESC
```

9. List the category ID and average units in stock for all categories whose products have an average units in stock less than 30. Put the list in order of average units in stock. Use the column alias "Average Stock" for average units in stock. (10 points)

```
SELECT Categories.CategoryID, AVG(Products.UnitsInStock)
AS AverageStock FROM categories, Products WHERE
Categories.CategoryID = Products.CategoryID AND
Average Stock < 30 GROUP BY Categories.CategoryID
ORDER BY Average Stock
```

-2

Name: David Norman

Section: \_\_\_\_\_

10. List the order ID, order date, product ID, product name, and customer (company) name for all orders that involve product IDs 1 or 3 or are from customers with the IDs 'AROUT', 'BONAP', or 'BOTTM'. Put the list in descending order of product name, then in ascending order of customer name. (10 points)

```
SELECT Orders.OrderID, Orders.OrderDate, Products.ProductID,
Products.ProductName, Customers.CompanyName FROM
[Order Details], Orders, Products, Customers WHERE
[Order Details].OrderID=Orders.OrderID AND
[Order Details].ProductID=Products.ProductID AND
Orders.CustomerID=Customers.CustomerID AND
(Products.ProductID IN (1,3) OR Customers.CustomerID
IN ('AROUT', 'BONAP', 'BOTTM')) ORDER BY
Products.ProductName DESC, Customers.CustomerName
```

11. List the product ID, product name, and units in stock for all products with more units in stock than the average units in stock for all products. Put the list in order of units in stock, then product name. (10 points)

```
SELECT Outer.ProductID, Outer.ProductName,
Outer.UnitsInStock FROM Products Outer WHERE
Outer.UnitsInStock > (SELECT AVG(Inner.UnitsInStock)
FROM Products Inner)
ORDER BY Outer.UnitsInStock, Outer.ProductName
```

Name: David Norman Section: \_\_\_\_\_

12. List the IDs and names of **all** customers, along with the IDs and dates of any orders they have placed. **Your results must list all customers, even those who haven't placed any orders.** Put your list in order of company name. (10 points)

```
SELECT Customers.Customer ID, Customers.Contact Name,  
Orders.Order ID, Orders.Order Date FROM Orders,  
Customers WHERE Customers.Customer ID =  
Orders.Customer ID ORDER BY Customers.Company Name
```

13. Write a query that answers the following question. How many **different** customers have placed orders? (5 points)

```
SELECT COUNT(DISTINCTUNIQUE(Customer ID)) FROM  
Orders
```

2

Name: David Norman Section: \_\_\_\_\_

14. List the order ID, product ID, unit price, quantity and extended price (unit price X quantity) for all orders that include products with the ID 1, 3 or 5. Put the list in order of product ID, then extended price. (Hint: This is a single-table query.) (5 points)

```
SELECT OrderID, ProductID, UnitPrice, Quantity,  
(UnitPrice * Quantity) AS Extended FROM  
[Order Details] WHERE ProductID IN (1,3,5)  
ORDER BY ProductID, Extended
```

15. List the product name, supplier's company name, and unit price for all products that have a unit price greater than the average unit price for all products supplied by that supplier. Put the list in order of supplier's company name, then descending order of unit price. (15 points)

```
SELECT OuterProducts.ProductName, OuterSuppliers.CompanyName,  
OuterProducts.UnitPrice FROM suppliers OuterSuppliers,  
Products OuterProducts WHERE OuterProducts.SupplierID = OuterSuppliers.SupplierID  
AND OuterProducts.UnitPrice > NO SPACES  
(SELECT AVG(InnerProducts.UnitPrice) FROM  
Products InnerProducts WHERE  
InnerProducts.SupplierID = OuterProducts.SupplierID)  
ORDER BY Suppliers.CompanyName, Products.UnitPrice  
DESC
```