

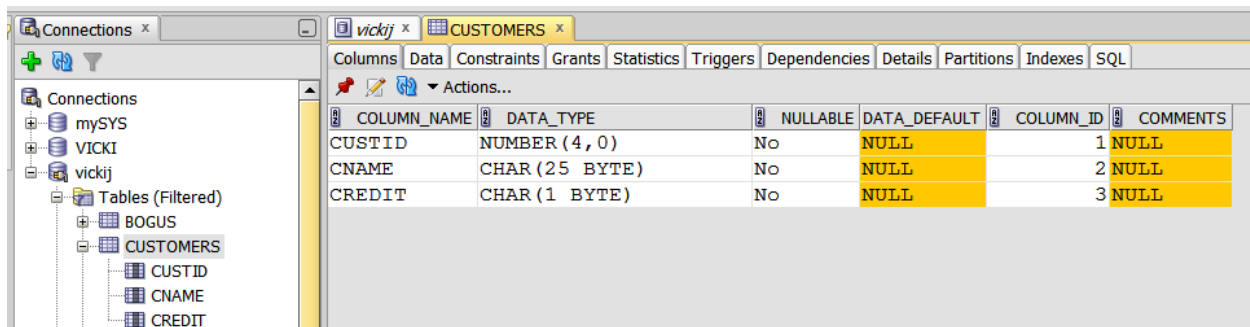
CIS 276 Lab 3 (50 points)

Introduction

This lab is to be done using PCC's Oracle database server at cis233s.pcc.edu, so you need to get the Oracle client software (SQL Developer) before you can work on the assignment. The purpose of this lab is simply to create the environment you will need in future labs. Specifically, you will build the Oracle command files to create and populate your own private **SalesDB** database. This is a fairly simple task, and provides you the opportunity to work with DDL statements and Oracle command files. The hard part of this lab will probably be getting your connection set up cis233s.pcc.edu and translating your SQL Server SQL into Oracle SQL.

You will need to use the form at <http://cis233s.pcc.edu:7777/pls/apex/f?p=127:1:469148269977242> to request a student account. When it has been created for you (it doesn't happen immediately) you will have the username and password to access your private area on the PCC Oracle server.

The data for your **SalesDB** tables is available in reference tables in the CIS276 database. **You should begin with a DESCRIBE on the CIS276 tables (DESC CIS276.CUSTOMERS;) or just look at SQL Developer's object explorer (or connections tab or navigator or whatever it's called).** Your metadata should conform to the information below (column_name, data type, nullable, and default value). It's fun to check out each tab too. Below is a picture of what I have for the SalesDB CUSTOMERS table:



The screenshot shows the SQL Developer interface with the 'CUSTOMERS' table selected in the 'Connections' tab. The table structure is displayed in the main pane, showing columns: CUSTID, CNAME, and CREDIT. The table is located in the 'vickij' schema under the 'Tables (Filtered)' view.

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
CUSTID	NUMBER (4, 0)	No	NULL	1	NULL
CNAME	CHAR (25 BYTE)	No	NULL	2	NULL
CREDIT	CHAR (1 BYTE)	No	NULL	3	NULL

Submission Notes

- Create a separate .sql file for each task as indicated on each question, and use the file names shown.
- Zip the ten .sql files into a file called **Lab3.zip**; notice this is a zip file and not an sql file.
- Submit your **Lab3.zip** file into the assignments area of the class website by the due date as shown in the class schedule and in the class calendar.
- You are responsible for knowing the due dates! Play it safe and submit at least 15 minutes before the time noted. You may submit early but late submissions, if accepted, immediately lose points.

Grading Notes

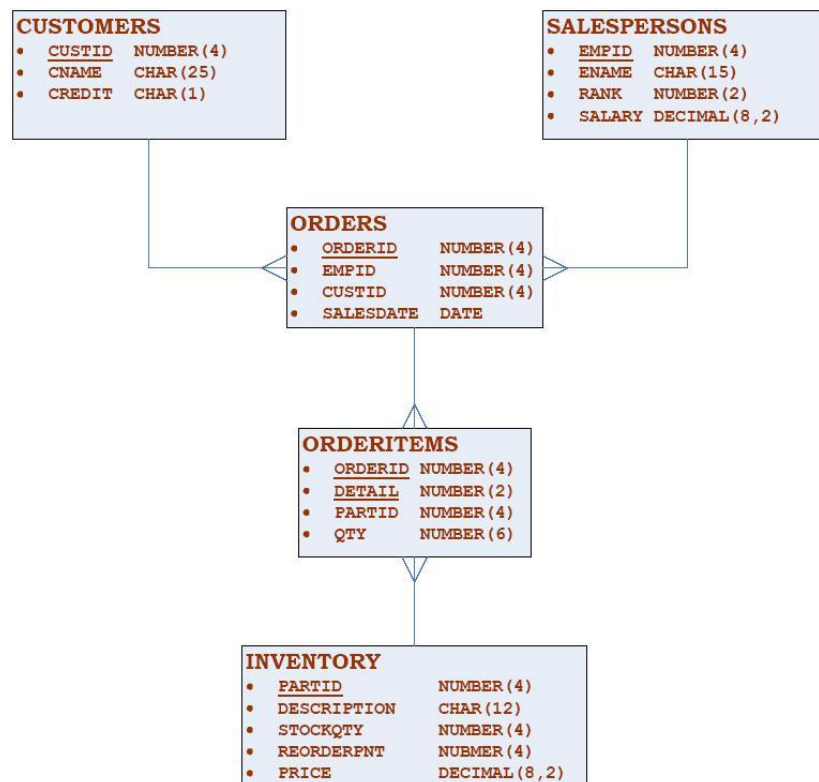
Remember - everything in this lab is Oracle, nothing is SQL Server. There are ten files containing either script or queries. **Each .sql file is worth five points.** Each .sql file will have a comment box containing at least your name and the question. Make sure you test your .sql files. Any .sql file that does not execute properly or produces an incorrect result will get zero (0) points. Please note that your .sql file does not need to contain anything other than the comment box at the top and the appropriate SQL DDL or DML

statements. Later in this course you will augment your SQL with procedural language components (error checking, variables, selection statements, loops, and so forth) but for this lab don't worry about writing programs - just write the queries.

You don't have to make your output look exactly like my output as long as you follow the instructions and display the correct data in (class) standard format. The answer file will be posted soon after the due date. A grading script will be used to test your command files. Attempts may be awarded some points so it is better to try than to do nothing. Please submit an *.sql file for each of the ten questions. At the minimum each file is to contain a comment box. You may copy/paste the questions from this document you're your comment box. Please use comments to show your assumptions however it is better to ask so there is no assumption in the first place.

Oracle Client Software Database Design for CIS276

The diagram below shows the tables, attributes, and relationships that you need to use for your **SalesDB** tables. The primary keys are underlined. (Note: the existing SalesDB may not conform to this information.)



- **CUSTOMERS**
 - **CUSTID** is the customer number and primary key of **CUSTOMERS**
 - **CNAME** is the customer name
 - **CREDIT** is the customer's credit rating; value must be "A", "B", or "C"
- **SALESPERSONS**
 - **EMPID** is the employee number and primary key of **SALESPERSONS**
 - **ENAME** is the employee name
 - **RANK** is the employee rank within the company; value must be 1, 2, or 3
 - **SALARY** is the employee salary; default value is \$1,000.00; value must be greater than or equal to \$1,000.00

- **ORDERS**
 - **ORDERID** is the order number and primary key of **ORDERS**
 - **EMPID** is the employee number of the salesperson who booked the order and is a foreign key to **SALESPERSONS**
 - **CUSTID** is the customer number of the customer who placed the order and is a foreign key to **CUSTOMERS**
 - **SALESDATE** is the date the order was placed; default value is **SYSDATE**
- **ORDERITEMS**
 - **ORDERID** is the number of the order associated with this line item; it is the primary key field for **ORDERITEMS** and the foreign key to **ORDERS**
 - **DETAIL** is the line number of this line item and is a sequential number starting at 1. Do not code for this auto-numbering scheme, however, as the detail value will be incremented later in our transaction processing.
 - **ORDERITEM** rows that share a common value for **ORDERID** (i.e. each line item for a given order) are distinguished by having a unique **DETAIL** value (i.e. each order line item has a unique line number). This is the composite primary key field.
 - **PARTID** is the part number that has been ordered and is a foreign key to **INVENTORY**
 - **QTY** is the quantity of the specified part that has been ordered
- **INVENTORY**
 - **PARTID** is the part number and primary key for **INVENTORY**
 - **DESCRIPTION** is the part name or description
 - **STOCKQTY** is the quantity of this part that we have in our inventory
 - **REORDERPNT** is the "threshold" for re-ordering the part from the supplier. If **STOCKQTY** is less than **REORDERPNT** then we should re-order the part. However this is not a constraint when creating the table so please do not make it one.
 - **PRICE** is the sales price of the part

Lab Assignment 3

1. Build the table create script in a file named **SalesDB_Create.sql**
 - Instructions: Create an .sql file that contains the **CREATE TABLE** statements to build the **SalesDB** tables. Include the primary keys, foreign keys, and domain constraints shown in the Database Design section (above) in your **CREATE TABLE** statements. The order of statements is important.
2. Write the drop tables SQL in a file named **SalesDB_Drop.sql**
 - Instructions: Create an .sql file that contains **DROP TABLE** statements to delete the **SalesDB** tables. The order of statements is important.
3. Three SQL statements will go into your file named **SalesDB_IndexView.sql**
 - Write a **CREATE INDEX** statement for each the following columns:
 - **ORDERS.salesdate** (do not make this a UNIQUE index); name it OrdersDateIdx.
 - **INVENTORY.description** (make this a UNIQUE index); name it InventoryDescriptionIdx.
 - Write a **CREATE VIEW** statement to project everything for an inner join of **ORDERS** to **ORDERITEMS** and call the view **ORDERS_ITEMS_VIEW**. Do not use * for the projection.
4. Build the table load script in a file named **SalesDB_Load.sql**
 - Instructions: Create an .sql file that contains **INSERT** statements to load the **SalesDB** tables from the corresponding CIS276 tables. Each table will have one **INSERT** statement.

5. Who earns less than or equal to \$2,000?
 - File to create: `q5.sql`
 - Columns to display: `SALESPERSONS.ename`, `SALESPERSONS.salary`
 - Instructions: Display the name and salary of all salespersons whose salary is less than or equal to \$2,000. Sort projection on salary high to low.
6. Which parts cost between one and seventeen dollars (inclusive)?
 - File to create: `q6.sql`
 - Columns to display: `INVENTORY.partid`, `INVENTORY.description`, `INVENTORY.price`
 - Instructions: Display the part id, description, and price of all parts where the price is between \$1.00 and \$17.00. Show the output in descending order of price. Use the `BETWEEN` clause.
7. Which part descriptions begin with the letter G (or g)? Code for case sensitivity.
 - File to create: `q7.sql`
 - Columns to display: `INVENTORY.partid`, `INVENTORY.description`
 - Instructions: Display the part id and description of all parts where the description begins with the capital letter G or the lower case letter g. Show the output in descending order of price.
8. What are the highest and lowest priced parts? One query please.
 - File to create: `q8.sql`
 - Columns to display: `INVENTORY.partid`, `INVENTORY.description`, `INVENTORY.price`
 - Instructions: Display the part id, description, and price for the highest and lowest priced parts in our inventory. You may include a column containing the literal value "highest" or "lowest".
9. Which sales people have NOT sold anything? (The correlated sub-query version)
 - File to create: `q9.sql`
 - Columns to display: `SALESPERSONS.ename`
 - Instructions: Display the employees that are not involved with an order, i.e. where the empid of the sales person does not appear in the `ORDERS` table. Display the names in alphabetical order. *Do not use joins at all - use sub-queries only.* Suggestion: use the EXISTS clause.
10. Which sales people have NOT sold anything? (The JOIN version)
 - File to create: `q10.sql`
 - Columns to display: `SALESPERSONS.ename`
 - Instructions: Display the employees that are **not** involved with an order, i.e. where the empid of the sales person does not appear in the `ORDERS` table. Display the names in alphabetical order. *Do not use sub-queries at all - use only JOINS only.*