SQL & PL/SQL

The objective of the course is to familiarize students with basics of ORACLE DBA (Database Architecture). Students will gain a conceptual understanding of the Oracle database architecture and how its components work and interact with one another. Students will also learn how to create an operational database and properly manage the various structures in an effective and efficient manner including performance monitoring, database security, user management, and backup/recovery techniques.

OUTLINE OF THE COURSE

S. No.	Topic	Minimum number of Hours
1	Introduction of Database, DB Design, RDBMS	5
2	ORACLE Versions, Installation and Architecture	2
3	SQL	15
4	Database Transactions and Security	5
5	PL/SQL	20
6	Interview FAQs	3

LECTURE: 50 hrs
PRACTICLE/TUTORIAL: 40 hrs
PROJECT: 30 hrs
TOTAL: 120 hrs

DETAILED SYLLABUS

1. Introduction of Database, DB Design, RDBMS:

Definition of Database, Definition of DBMS FMS (File Management Systems), Advantages of DBMS over FMS, DB Life Cycle, DB Design (ER Model, Normalization), Data Models, Feature of RDBMS, CODD Rules, Client/Server Architecture

2. ORACLE

Oracle Versions, Oracle11g Installation, Oracle Architecture

3. SQL

Introduction to SQL, SQL Standards, SQL Sublanguages, About SQL*PLUS, Differentiate between SQL AND SQL*PLUS, Data types in ORACLE, Operators in ORACLE, Schemas and Schemas object, Table Creation, Inserting Data, Data Retrieval (SELECT Statement, Column alias, Table alias, Clauses in SQL), Data Filtering (WHERE clause, ORDER BY clause), DML Commands (INSERT, DELETE, UPDATE, MERGE, INSERT ALL, Copy data from one table to another), DDL Commands, Integrity Constraints, Built-in Functions, Data Aggregation, JOIN Operations, SET Operators, Sub Queries

4. Database Transactions and Security

Commit Command, Rollback command, Save point command, System privileges and Objects, privileges, Granting and revoking privileges, Creating users and roles, Schema Objects, About locks and Isolation Levels, About table and partitioning, Hierarchical Queries

5. PL/SQL

Introduction to PL/SQL, Advantages of PL/SQL, PL/SQL Architecture, Data types in PL/SQL, PL/SQL Program Structure, Simple PL/SQL Programs, Embedding SQL statements in PL/SQL, Conditional Statements, Loops, Cursors, Exception Handling, Subprograms, Stored Procedures, Function and Packages, Triggers, Collections in PL/SQL, Built-in packages in PL/SQL, ORACLE 10g features, ORACLE 11g features

6. Interview FAQs

REFERENCE BOOKS:

1.Oracle 11g: PL/SQL Programming by Joan Casteel

ONLINE RESOURCES

www.tutorialspoint.com/plsql/index.htm

https://docs.oracle.com/cd/B19306_01/server.102/b14357/qstart.htm

http://www.w3schools.com/sql/default.asp

Assignments

1. Create a query to display all the data from the *Employees* table.

2. The following SELECT statement executes successfully (True / False)

```
FROM employees
SELECT last_name, first_name
```

3. Create a query to display the department number, department name, and manager number. Name the last column (manager number) heading as “MNG” (*Employees* table).

4. The following SELECT statement executes successfully (True / False)

```
SELECT department_name, department_name FROM
departments
```

5. Create a query to display the employee number, first name, last name, phone number and department number (*Employees* table).
6. Create a query to display the first name, last name, hire date, salary, and salary after a raise of 20%. Name the last column (salary after a raise) heading as “ANNUAL_SAL” (*Employees*table)
7. Create a query to display the last name concatenated with the first name, separated by space, and the telephone number concatenated with the email address, separated by hyphen. Name the column headings “FULL_NAME” and “CONTACT_DETAILS” respectively (*Employees* tables).
8. Create a query to display the last name concatenated with *job_id* column, separated by space. Name this column heading as “EMPLOYEE_AND_TITLE” (*Employees* table).
9. Create a query to display the first name, last name, salary, and hire date concatenated with the literal string “HD”, separated by space. Name the column headings “FN”, “LN”, “SAL”, and “HD” respectively (*Employees* table).
10. Create a query to display the unique salaries in *Employees* tables.
11. Create a query to display the unique combination of values in *department_id* and *job_id*columns (*Employees* table).
12. Display the first name concatenated with the last name, separated by comma, and salary, for all employees whose salary not in the range between 7000 and 15000. Sort the query in ascending order by the full name (*Employees* table).

13. Display the first name concatenated with the last name, separated by comma, the phone number concatenated with the email address, separated by hyphen, and salary, for all employees whose salary is in the range of 5000 and 10000. Name the column headings:
"FULL_NAME", "CONTACTS" and "SAL" respectively (*Employees* table).

14. Display all data from *Employees* table for all employees whose:
salary is in the range of 6000 and 800 **and** their commission is not null **or** department number is not equal to 80, 90 and 100 **and** their hire date is before January 1st, 1990.

15. Display last name, job id and hire date for all employees who was hired during December 12th, 1995 and April 17th, 1998.

16. Display the first name concatenated with last name, hire date, commission percentage, telephone, and salary for all employees whose salary is greater than 10000 **or** the third digit in their phone number equals 5. Sort the query in a descending order by the first name (*Employees* table).

17. Display the last name and salary for all employees who earn more than 12000 (*Employees* table).

18. Display the last name and department number for all employees whose department number is equal to 50 or 80. Perform this exercise once by using the IN operator, once by using the OR operator.

19. Display the first name and salary for all employees who doesn't earn any commission.

20. Display the first name, salary, and manager number for all employees whose manager number is not null.

21. Display the first name concatenated with the last name (separated by space), the main phone number concatenated with street (separated by space), and monthly discount for all customers whose monthly discount is in the range between 11 and 27. Name the column headings FULL_NAME, CONTACTS, and DC respectively (*Customers* table).

22. Display all data from *Customers* table for :
    1. All customers who live in New York **and** whose monthly discount is in the range between 30 and 40 **or**
    2. All customers whose package number is not 8,19, or 30 **and** whose join date is before January 1st, 2007

23. Display the last name, package number, and birthdate for all customers whose join date is in the range between *December 12th, 2007* and *April 17th, 2010* (*Customers* table).

24. Display the package number, start date, and speed for all packages whose start date is before January 1st, 2007 (*Packages* table)
25. Display the package number, speed, and sector number for all packages whose sector number equals 1 (*Packages* table).
26. Display the package number, speed and sector number for all packages with internet speed of “5Mbps” or “10Mbps” (*Packages* table).
27. Display the last name and monthly discount for all customers live in Orlando (Customers table).
28. Display the last name and package number for all customers whose package number equals 9 or 18. Preform the exercise once using IN operator, once using OR (*Customers* table).
29. Display the first name, main phone number and secondary phone number for all customers without a secondary phone number (secondary phone number is null).
30. Display the first name, monthly discount and package id for all customers without any monthly discount (monthly discount is null).
31. Average salary per department
    1. Display the department number and average salary for each department.
    2. Modify your query to display the results only for departments 50 or 80.
32. Numer of employees per job id
    1. Display the job id and the number of employees for each job id.
    2. Modify your query to display the results only for employees whose salary is greater the 10000.
    3. Modify your query again, this time display the results only for jobs with more than 2 people.
33. Display the department number, job id, and the average salary for each department and job id.
34. Managers and highest salary
    1. Display the manager number and the highest salary for each manager number.
    2. Modify your query to display the results only for employees whose salary is greater than 10000.

35. Display the job id and minimum salary for each job id, for all jobs whose minimum salary is greater than 7000.

36. Display the department number, and the average salary for each department, for all departments whose number is in the range of 20 and 80, and their average salary is greater than 9000.

37. Package number and average monthly discount (*Customers* table) –

    1. Display the package number and the average monthly discount for each package.

    2. Display the package number and the average monthly discount for each package, only for packages whose number equals 22 or 13.

38. Display the highest, lowest and average monthly payment for each internet speed (*Packages* table).

39. The number of customer in each internet package (*Customers* table) –

    1. Display the package number and the number of customers for each package number.
    2. Modify the query to display the package number and number of customers for each package number, only for the customers whose monthly discount is greater than 20.
    3. Modify the query to display the package number and number of customers for each package number, only for the packages with more than 100 customers.

40. Display the state, city and number of customers for each state and city.

41. Cities and the average monthly discount (*Customers* table) –

    1. Display the city and the average monthly discount for each city

    2. Display the city and the average monthly discount for each city, only for the customers whose monthly discount is greater than 20

42. States and the lowest monthly discount (*Customers* table) –
    1. Display the state and the lowest monthly discount for each state.

    2. Display the state and lowest monthly discount for each state, only for states where the lowest monthly discount is greater than 10

43. Display the internet speed and number of package for each internet speed, only for the internet speeds with more than 8 packages.

44. **Conditional logic**:-Rewrite the following IF statements so that you do *not* use an IF statement to set the value of no_revenue . What is the difference between these two statements? How does that difference affect your answer?

```
1.    IF total_sales <= 0
2.    THEN
3.    no_revenue := TRUE;
4.    ELSE
5.    no_revenue := FALSE; 6. END IF; 7.
8.       IF total_sales <= 0
9.       THEN
10.      no_revenue := TRUE;
11.      ELSIF total_sales > 0
12.      THEN
13.      no_revenue := FALSE;
```

```
END IF;
```

45. Which procedure will *never* be executed in this IF statement?

```
46.      IF (order_date > SYSDATE) AND order_total >= min_order_total
47.      THEN
48.      fill_order (order_id, 'HIGH PRIORITY');
49.      ELSIF (order_date < SYSDATE) OR
50.      (order_date = SYSDATE)
51.      THEN
52.      fill_order (order_id, 'LOW PRIORITY');
53.      ELSIF order_date <= SYSDATE AND order_total < min_order_total
54.      THEN
55.      queue_order_for_addtl_parts (order_id);
56.      ELSIF order_total = 0
57.      THEN
58.      disp_message (' No items have been placed in this order!');
59.      END IF;
```

60. **Loops** :- How many times does the following loop execute?

```
61.      FOR year_index IN REVERSE 12 .. 1
62.      LOOP
63.      calc_sales (year_index);
         END LOOP:
```

64. Select the type of loop (FOR, WHILE, simple) appropriate to meet each of the following requirements:
    1. Set the status of each company whose company IDs are stored in a PL/SQL table to closed.
    2. For each of twenty years in the loan-processing cycle, calculate the outstanding loan balance for the specified customer. If the customer is a preferred vendor, stop the calculations after twelve years.

3. Display the name and address of each employee returned by the cursor.
4. Scan through the list of employees in the PL/SQL table, keeping count of all salaries greater than $50,000. Don't even start the scan, though, if the table is empty or if today is a Saturday or if the first employee in the PL/SQL table is the president of the company.

65. Identify the problems with (or areas for improvement in) the following loops. How would you change the loop to improve it?

```
1.  FOR i IN 1 .. 100
2.  LOOP
3.  calc_totals (i);
4.  IF i > 75
5.  THEN
6.  EXIT;
7.  END IF;
8.  END LOOP;
```

9.

```
10.  OPEN emp_cur;
11.  FETCH emp_cur INTO emp_rec;
12.  WHILE emp_cur%FOUND
13.  LOOP
14.    calc_totals (emp_rec.salary);
15.    FETCH emp_cur INTO emp_rec;
16.    EXIT WHEN emp_rec.salary > 100000;
17.  END LOOP;
18.  CLOSE emp_cur;
```

19.

```
20.  FOR a_counter IN lo_val .. hi_val
21.  LOOP
22.    IF a_counter > lo_val * 2
23.    THEN
24.      hi_val := lo_val;
25.    END IF;
26.  END LOOP;
```

27.

```
28.  DECLARE
29.    CURSOR emp_cur IS SELECT salary FROM emp;
30.    emp_rec emp_cur%ROWTYPE
```

```
31.  BEGIN
32.    OPEN emp_cur;
33.    LOOP
34.      FETCH emp_cur INTO emp_rec;
35.      EXIT WHEN emp_cur%NOTFOUND;
36.      calc_totals (emp_rec.salary);
37.    END LOOP;
```

```
38.     CLOSE emp_cur;
39.   END;
```

40.

```
41.   WHILE no_more_data
42.   LOOP
43.     read_next_line (text);
44.     no_more_data := text IS NULL;
45.     EXIT WHEN no_more_data;
46.   END LOOP;
```

47.

```
48.   FOR month_index IN 1 .. 12
49.   LOOP
50.     UPDATE monthly_sales
51.       SET pct_of_sales = 100
52.      WHERE company_id = 10006
53.       AND month_number = month_index;
54.   END LOOP;
```

55.

```
56.   DECLARE
57.     CURSOR emp_cur IS SELECT ... ;
58.   BEGIN
59.     FOR emp_rec IN emp_cur
60.     LOOP
61.       calc_totals (emp_rec.salary);
62.     END LOOP;
63.     IF emp_rec.salary < 10000
64.     THEN
65.       DBMS_OUTPUT.PUT_LINE ('Give "em a raise!');
66.     END IF;
67.     CLOSE emp_cur;
68.   END;
```

69.

```
70.                     DECLARE
71.                     CURSOR checked_out_cur IS
72.                     SELECT pet_id, name, checkout_date
73.                     FROM occupancy
74.                     WHERE checkout_date IS NOT NULL;
75.                     BEGIN
76.                     FOR checked_out_rec IN checked_out_cur
77.                     LOOP
78.                     INSERT INTO occupancy_history (pet_id, name,
                        checkout_date)
79.                     VALUES (checked_out_rec.pet_id,
80.                     checked_out_rec.name,
81.                     checked_out_rec.checkout_date);
```

```
82.     END LOOP;
83.   END;
```

66. How many times does the following WHILE loop execute?

```
67.  DECLARE
68.  end_of_analysis BOOLEAN := FALSE;
69.  CURSOR analysis_cursor IS SELECT ...;
70.  analysis_rec analysis_cursor%ROWTYPE;
71.  next_analysis_step NUMBER;
72.  PROCEDURE get_next_record (step_out OUT NUMBER) IS
73.  BEGIN
74.  FETCH analysis_cursor INTO analysis_rec;
75.  IF analysis_rec.status = 'FINAL'
76.  THEN
77.  step_out := 1; 78.      ELSE
79.          step_out := 0;
80.          END IF;
81.          END;
82.          BEGIN
83.          OPEN analysis_cursor;
84.          WHILE NOT end_of_analysis
85.          LOOP
86.          get_next_record (next_analysis_step);
87.          IF analysis_cursor%NOTFOUND AND
88.          next_analysis_step IS NULL
89.          THEN
90.          end_of_analysis := TRUE;
91.          ELSE
92.          perform_analysis;
93.          END IF;
94.          END LOOP;
          END;
```

In each of the following PL/SQL blocks, a VALUE_ERROR exception is raised (usually by an attempt to place too large a value into a local variable). Identify which exception handler (if any -- the exception could also go unhandled) will handle the exception by writing down the message that will be displayed by the call to PUT_LINE in the exception handler. Explain your choice.

```
95.  DECLARE
96.  string_of_5_chars VARCHAR2(5);
97.  BEGIN
98.  string_of_5_chars := 'Steven';
99.  END;
```

100.

```
101.        DECLARE
102.        string_of_5_chars VARCHAR2(5);
103.        BEGIN
104.        BEGIN
105.        string_of_5_chars := 'Steven';
```

```
106.        EXCEPTION
107.        WHEN VALUE_ERROR
108.        THEN
109.        DBMS_OUTPUT.PUT_LINE ('Inner block');
110.        END;
111.        EXCEPTION
112.        WHEN VALUE_ERROR
113.        THEN
114.        DBMS_OUTPUT.PUT_LINE ('Outer block'); 115. END;
```

116.

```
117.        DECLARE
118.        string_of_5_chars VARCHAR2(5) := 'Eli';
119.        BEGIN
120.        BEGIN
121.        string_of_5_chars := 'Steven';
122.        EXCEPTION
123.        WHEN VALUE_ERROR
124.        THEN
125.        DBMS_OUTPUT.PUT_LINE ('Inner block');
126.        END;
127.        EXCEPTION
128.        WHEN VALUE_ERROR
129.        THEN DBMS_OUTPUT.PUT_LINE ('Outer block'); 130. END;
```

131.

```
132.    DECLARE
133.    string_of_5_chars VARCHAR2(5) := 'Eli'; 134. BEGIN
135.    DECLARE
136.    string_of_3_chars VARCHAR2(3) := 'Chris';
137.    BEGIN
138.    string_of_5_chars := 'Veva';
139.    EXCEPTION
140.    WHEN VALUE_ERROR
141.    THEN DBMS_OUTPUT.PUT_LINE ('Inner block');
142.    END;
143.    EXCEPTION
144.    WHEN VALUE_ERROR
145.    THEN DBMS_OUTPUT.PUT_LINE ('Outer block'); 146. END;
```

147.

```
148.        DECLARE
149.        string_of_5_chars VARCHAR2(5);
150.        BEGIN
151.        BEGIN
152.        string_of_5_chars := 'Steven';
153.        EXCEPTION
```

```
154.    WHEN VALUE_ERROR
155.    THEN
156.    RAISE NO_DATA_FOUND;
157.    WHEN NO_DATA_FOUND
158.    THEN
159.    DBMS_OUTPUT.PUT_LINE ('Inner block');
160.    END;
161.    EXCEPTION
162.    WHEN NO_DATA_FOUND
163.    THEN
164.    DBMS_OUTPUT.PUT_LINE ('Outer block');
165.    END;
```

Overview

In this assignment you will build a sample database. You will create the tables in Oracle, add sample data and create several queries and a report.

Activities

Step 1.

Using Oracle, create the tables identified below in the following order:

DATABASE TABLES:

Campus (CampusID, CampusName, Street, City, State, Zip, Phone, CampusDiscount)

Position (PositionID, Position, YearlyMembershipFee)

Members (MemberID, LastName, FirstName, CampusAddress, CampusPhone, CampusID, PositionID, ContractDuration)
FK CampusID --> Campus(CampusID)
PositionID --> Position(PositionID)

Prices (FoodItemTypeID, MealType, MealPrice)

FoodItems (FoodItemID, FoodItemName, FoodItemTypeID)
FK FoodItemTypeID --> Prices(FoodItemTypeID)

Orders (OrderID, MemberID, OrderDate)
FK MemberID --> Members(MemberID)

OrderLine (OrderID, FoodItemsID, Quantity)
FK OrderID --> Orders(OrderID)
FoodItemsID --> FoodItems(FoodItemID) **STRUCTURE NOTES**:

- Use the proper naming convention for your constraints:
  Example: Constraint TableName_FieldName_ConstraintID
  (Campus_CampusID_PK)
- Set up the Primary Keys for each table with Constraints listed. **Note**: The OrderLine Table has a composite Primary Key
- Add Your Foreign Keys for each table with your Constraints listed.
- Set up your Sequence for the Prices table ONLY. Remember to follow the proper naming convention. The Sequence will be used in the insert commands to add your auto numbering into the Primary Key (FoodItemTypeID) fields. Name the Sequence "Prices_FoodItemID_Seq"
- Make the Data Types for all the Primary Keys and their corresponding Foreign Keys Varchar2(5).
- Make the Data Type for OrderDate Varchar2(25). (we won't worry about the date format, way too complicated for what we are doing).
- Make the Data Types for the MealPrice and YearlyMembershipFee Decimal, 7 digits maximum with 2 digits to the right of the decimal place, so that we can perform calculations on them.
- Make the Data Types for ContractDuration, and Quantity Integer with 3 digits maximum for calculation purposes.
- Make the Data Type for CampusDiscount Decimal, 2 digits maximum with 2 digits to the right of the decimal place.

Step 2.

Use the Insert Into Command to add your data to each table. Add data to your primary tables first and then to your secondary tables. Also, remember to use the sequence code with your insert statement to add the auto number value to each primary key field.

DATA TO BE INSERTED:

Campus:
'1','IUPUI','425 University Blvd.','Indianapolis', 'IN','46202', '317-274-4591',.08
'2','Indiana University','107 S. Indiana Ave.','Bloomington', 'IN','47405', '812-8554848',.07
'3','Purdue University','475 Stadium Mall Drive','West Lafayette', 'IN','47907', '765-4941776',.06

Position:
'1','Lecturer', 1050.50
'2','Associate Professor', 900.50
'3','Assistant Professor', 875.50
'4','Professor', 700.75
'5','Full Professor', 500.50

Members:
'1','Ellen','Monk','009 Purnell', '812-123-1234', '2', '5', 12
'2','Joe','Brady','008 Statford Hall', '765-234-2345', '3', '2', 10
'3','Dave','Davidson','007 Purnell', '812-345-3456', '2', '3', 10
'4','Sebastian','Cole','210 Rutherford Hall', '765-234-2345', '3', '5', 10
'5','Michael','Doo','66C Peobody', '812-548-8956', '2', '1', 10
'6','Jerome','Clark','SL 220', '317-274-9766', '1', '1', 12
'7','Bob','House','ET 329', '317-278-9098', '1', '4', 10
'8','Bridget','Stanley','SI 234', '317-274-5678', '1', '1', 12
'9','Bradley','Wilson','334 Statford Hall', '765-258-2567', '3', '2', 10

Prices: **Note - Remember that these Primary Key Values should be entered using the Sequence (autonumber)**
'1','Beer/Wine', 5.50
'2','Dessert', 2.75
'3','Dinner', 15.50
'4','Soft Drink', 2.50 '5','Lunch', 7.25

FoodItems:
'10001','Lager', '1'
'10002','Red Wine', '1'
'10003','White Wine', '1'
'10004','Coke', '4'
'10005','Coffee', '4'
'10006','Chicken a la King', '3'
'10007','Rib Steak', '3'
'10008','Fish and Chips', '3'
'10009','Veggie Delight', '3'
'10010','Chocolate Mousse', '2'
'10011','Carrot Cake', '2'
'10012','Fruit Cup', '2'
'10013','Fish and Chips', '5'
'10014','Angus Beef Burger', '5'
'10015','Cobb Salad', '5'

Orders:
'1', '9', 'March 5, 2005'
'2', '8', 'March 5, 2005'
'3', '7', 'March 5, 2005'
'4', '6', 'March 7, 2005'
'5', '5', 'March 7, 2005'
'6', '4', 'March 10, 2005'
'7', '3', 'March 11, 2005'
'8', '2', 'March 12, 2005'
'9', '1', 'March 13, 2005'

OrderLine:
'1','10001',1
'1','10006',1
'1','10012',1
'2','10004',2
'2','10013',1
'2','10014',1
'3','10005',1
'3','10011',1
'4','10005',2
'4','10004',2
'4','10006',1
'4','10007',1
'4','10010',2
'5','10003',1
'6','10002',2
'7','10005',2
'8','10005',1
'8','10011',1
'9','10001',1

Step 3.

Create the queries listed below:

NOTE: Before you begin to run Queries 1 - 8, Enter the following two commands at the SQL Prompt:

Set Linesize 110 Set
Pagesize 90

This will allow you to see most, if not all, of your data without it wrapping.

1. Select all records from each table
2. List all of your constraints in the database
3. List all of your table names in the database
4. List your sequence name in the database **select sequence_name from user_sequences;**
5. List the Columns and Datatypes of each table
6. Create a listing of all Faculty Members (First and Last), their Faculty Position and the University that they are affiliated with (Name), along with their Monthly_Dues (Calculated Field with a column alias). Sort the records in descending order by University and then by Faculty's last name in ascending order.
7. Create a listing that shows the various food items that the faculty club serves (Name of the food item, type of food item and the price of the food item). Note: List no alcoholic beverages. Sort the records in ascending order by price.

8. List the OrderID, Order Date, Faculty Member's Name, Campus Name, each FoodItem that makes up a given order, the type of meal, cost of the meal, quantity ordered and the total line total (calculated field and column alias). Sort by Order IDs in descending order.

Step 4.

Create 1 formatted Report following the examples given in the SQL Book. The report should be similar to the following in format:

View Requirements:

- You will have to create a view that contains your report data. View Name: IFC_Report
- Sort your data in ascending order by the OrderID within your view.
- This view is similar in structure to Query 8 in Step 3.
- The grouping information within the report that does not repeat contains 5 fields concatenated together.
  Include Fields: OrderID, OrderDate, FirstName, LastName and CampusName)
- The last field of the report (Totals) is a calculated field. This field takes into consideration the following:
  Line Item total with the Campus Discount and tax of 6% figured in.

Report Requirements:

- Make sure that you list the report title and headings as listed on the report.
- Make sure that you set your formatting for each column in your report.
  Remember that there are different formats for Character and Number Datatypes.
- Sums (totals) are calculated for each order grouping (info that does not repeat) and for the entire report.
- Remember that the linesize, pagesize, amount of characters that you allot to each field's format and the number of lines that you skip for your grouping will all affect your report's outputted display.

In the final text file that you will submit, name your queries Query 1 through Query 8, matching the criteria above. **NOTE**: Query 1 would be labeled Query1a, Query1b, Query1c depending on the number of tables that you have in your database.