# Oracle for Developers
# (DBMS SQL)
# Lab Book

# Document Revision History

| Date | Revision No. | Author | Summary of Changes |
|---|---|---|---|
| 05-Feb-2009 | 0.1D | Rajita Dhumal | Content Creation |
| 09-Feb-2009 |  | CLS team | Review |
| 02- Jun-2011 | 2.0 | Anu Mitra | Integration Refinements |
| 30-Nov-2012 | 3.0 | Karthikeyan R | Revamp of lab assignments |
| 26-Feb-2015 | 4.0 | Kavita Arora | Rearranging of lab assignments |
| May 2016 | 5.0 | Kavita Arora | Lab Refinements |

# Table of Contents

## Contents

## Getting Started

### Overview

This lab book is a guided tour for learning DBMS SQL. It comprises ‘To Do’ assignments. Follow the steps provided and work out the ‘To Do’ assignments.

### Setup Checklist for DBMS SQL

Here is what is expected on your machine in order for the lab to work.

### Minimum System Requirements

- Intel Pentium 90 or higher (P166 recommended)
- Microsoft Windows 95, 98, or NT 4.0, 2k, XP.
- Memory: 32MB of RAM (64MB or more recommended)

### Please ensure that the following is done:

- Oracle Client installed.
- Connectivity to Oracle Server

### Instructions

- For all coding standards refer Appendix A. All lab assignments should refer coding standards.
- Create a directory by your name in drive <drive>. In this directory, create a subdirectory <DBMS SQL>_assgn. For each lab exercise create a directory as lab <lab number>.

### Learning More (Bibliography if applicable)

NA

## Problem Statement / Case Study

**Table descriptions**

**Emp**

| Name | Null? | Type |
|---|---|---|
| Empno | Not Null | Number(4) |
| Ename | - | Varchar2(10) |
| job | | Varchar2(9) |
| mgr | | Number(4) |
| Hiredate | | Date |
| Sal | - | Number(7,2) |
| Comm | - | Number(7,2) |
| Deptno | - | Number(2) |

**Designation_Master**

| Name | Null? | Type |
|---|---|---|
| Design_code | Not Null | Number(3) |
| Design_name | | Varchar2(50) |

**Department_Master**

| Name | Null? | Type |
|---|---|---|
| Dept_Code | Not Null | Number(2) |
| Dept_name | | Varchar2(50) |

**Student_Master**

| Name | Null? | Type |
|---|---|---|
| Student_Code | Not Null | Number(6) |
| Student_name | Not Null | Varchar2(50) |
| Dept_Code | | Number(2) |
| Student_dob | | Date |
| Student_Address | | Varchar2(240) |

**Student_Marks**

| Name | Null? | Type |
|---|---|---|
| Student_Code | | Number(6) |
| Student_Year | Not Null | Number |
| Subject1 | | Number(3) |
| Subject2 | | Number(3) |
| Subject3 | | Number(3) |

**Staff_Master**

| Name | Null? | Type |
|---|---|---|
| Staff_code | Not Null | Number(8) |
| Staff_Name | Not Null | Varchar2(50) |
| Design_code | | Number |
| Dept_code | | Number |
| HireDate | | Date |
| Staff_dob | | Date |
| Staff_address | | Varchar2(240) |
| Mgr_code | | Number(8) |
| Staff_sal | | Number (10,2) |

**Book_Master**

| Name | Null? | Type |
|---|---|---|
| Book_Code | Not Null | Number(10) |
| Book_Name | Not Null | Varchar2(50) |
| Book_pub_year | | Number |
| Book_pub_author | Not Null | Varchar2(50) |

**Book_Transactions**

| Name | Null? | Type |
|---|---|---|
| Book_Code | | Number |
| Student_code | | Number |
| Staff_code | | Number |
| Book_Issue_date | Not Null | Date |
| Book_expected_return_date | Not Null | Date |
| Book_actual_return_date | | Date |

# Lab 1.Data Query Language

| | |
|---|---|
| **Goals** | • Query the database<br>• Usage of various operators in select statements |
| **Time** | 1 hr 30 min |

## 1.1 : Data Query Language

1. Retrieve the details (Name, Salary and dept code) of the staff who are working in department code 20, 30 and 40.
2. Display the code and total marks for every student. Total Marks will be calculated as subject1+subject2+subject3 .(Refer Student_marks table )
3. List the Name and Designation code of the staff who have joined before Jan 2003 and whose salary range is between 12000 and 25000. Display the columns with user defined Column headers. Hint: Use As clause along with other operators
4. List the code, name, and department number of the staff who have experience of 18 or more years and sort them based on their experience.
5. Display name concatenated with dept code separated by comma and space. Name the column as 'Student Info'.
6. Display the staff details who do not have manager. Hint: Use is null
7. Write a query which will display name, department code and date of birth of all students who were born between January 1, 1981 and March 31, 1983. Sort it based on date of birth (ascending).Hint: Use between operator
8. Display the Book details that were published during the period of 2001 to 2004. Also display book details with Book name having the character '&' anywhere.
9. Display the Book details where the records have the word "COMP" anywhere in the Book name.
10. List the names of the staff having '_' character in their name.

# Lab 2.Single Row and Group Functions

| | |
|---|---|
| **Goals** | • Querying tables using single row functions<br>• Querying tables using date functions<br>• Querying tables using number functions<br>• Querying tables using group functions |
| **Time** | 1 hr 45 min |

## 2.1 : Single Row Functions:

1. Create a query which will display Staff Name, Salary of each staff. Format the salary to be 15 character long and left padded with '$'.
2. Display name and date of birth of students where date of birth must be displayed in the format similar to "January, 12 1981" for those who were born on Saturday or Sunday.
3. Display the name and salary of the staff. Salary should be represented as X. Each X represents a 1000 in salary. Hint: Divide salary by 1000, use rpad to substitute an 'X' for every 1000.

   Sample Output

```
JOHN        10000       XXXXXXXXXX
ALLEN       12000       XXXXXXXXXXXX
```

4. List the details of the staff who have joined in first half of December month (irrespective of the year).
5. Write a query that displays Staff Name, Salary, and Grade of all staff. Grade depends on the following table.

| Salary | Grade |
|---|---|
| Salary >=50000 | A |
| Salary >= 25000 < 50000 | B |
| Salary>=10000 < 25000 | C |
| OTHERS | D |

6. Display the Staff Name, Hire date and day of the week on which staff was hired. Label the column as DAY. Order the result by the day of the week starting with Monday. Hint :Use to_char with hiredate and formats 'DY' and 'D'

7. Show staff names with the respective numbers of asterisk from Staff_Masters table except first and last characters. For example: KING will be replaced with K**G. .
   Hint: Use substring, rpad and length functions.
8. Write a query to find the position of third occurrence of 'i' in the given word 'Mississippi'.
9. Display Student code, Name and Dept Name. Display "Electricals" if dept code = 20, "Electronics" if Dept code =30 and "Others" for all other Dept codes in the Dept Name column. Hint : Use Decode

## 2.2 : Group Functions:

1. Display the Highest, Lowest, Total & Average salary of all staff. Label the columns Maximum, Minimum, Total and Average respectively for each Department code. Also round the result to the nearest whole number.

2. Display Department code and number of managers working in that department. Label the column as 'Total Number of Managers' for each department.

3. Get the Department number, and sum of Salary of all non managers where the sum is greater than 20000.

# Lab 3.JOINS AND SUBQUERIES

|  |  |
|---|---|
| **Goals** | • Querying multiple tables using joins<br>• Querying tables using subqueries<br>• Querying tables using set operators |
| **Time** | 2 hr 30 min |

### 3.1 : Joins and Subqueries

1. Write a query which displays Staff Name, Department Code, Department Name, and Salary for all staff who earns more than 20000.
2. Create a query that will display Student Code, Student Name, Department Name, Subject1, Subject2, and Subject3 for all students who are getting 60 and above in each subject from department 10 and 20.
3. Create a query that will display Staff Code, Staff Name, Department Name, Designation name, Book Code, Book Name, and Issue Date. For only those staff who have taken any book in last 30 days. . If required, make changes to the table to create such a scenario.
4. Display the unique list of Book code and Book name from the Book transaction.
5. List Staff Code, Staff Name, and Salary for those who are getting less than the average salary of organization.
6. List the Staff Code, Staff Name who are not Manager.
7. Display Author Name, Book Name for those authors who wrote more than one book.
8. Display top ten students for a specified department. Details are:

   Student Code, Student Name, Department Name, Subject1, Subject2, Subject3, Total.
9. List the details of the staff, experience (in years) whose designations are either PROFESSOR or LECTURER.
10. Create a query that will display the Staff Name, Department Name, and all the staff who work in the same department as a given staff. Give the column as appropriate label.
11. Display the Student Code, Student Name, and Department Name for that department in which there are maximum number of student are studying.
12. Display Staff Code, Staff Name, Department Name, and Designation name for those who have joined in last 3 months.
13. Write a query to display number of people in each Department. Output should display Department Code, Department Name and Number of People.

## Lab 4.Database Objects

| | |
|---|---|
| **Goals** | • Following set of questions are designed to implement the following concepts<br>• Creating Database objects like tables, views , etc<br>• Modifying Database objects<br>• Deleting Database objects<br>• Usage of Data Dictionary tables |
| **Time** | 4 hr 30 min |

### 4.1: Database Objects

1. Create the Customer table with the following columns.

| | |
|---|---|
| CustomerId | Number(5) |
| Cust_Name | varchar2(20) |
| Address1 | Varchar2(30) |
| Address2 | Varchar2(30) |

2. Modify the Customer table Cust_Name column of datatype with Varchar2(30), rename the column to CustomerName and it should not accept Nulls.
3. a) Add the following Columns to the Customer table.

| | |
|---|---|
| Gender | Varchar2(1) |
| Age | Number(3) |
| PhoneNo | Number(10) |

b) Rename the Customer table to Cust_Table

c) Insert rows with the following data in to the Customer table.

Insert into customer values: (1000, ‘Allen’, ‘#115 Chicago’, ‘#115 Chicago’, ‘M’, ‘25, 7878776’)

In similar manner, add the below records to the Customer table:

- 1000, Allen, #115 Chicago, #115 Chicago, M, 25, 7878776
- 1001, George, #116 France, #116 France, M, 25, 434524
- 1002, Becker, #114 New York, #114 New York, M, 45, 431525

4. Add the Primary key constraint for CustomerId with the name CustId_Prim.
5. a) Disable the constraint on CustomerId, and insert the following data:

- 1002, Becker, #114 New York, #114 New york , M, 45, 431525
- 1003, Nanapatekar, #115 India, #115 India , M, 45, 431525

b) Drop the constraint CustId_Prim on CustomerId and insert the following Data. Alter Customer table, drop constraint Custid_Prim.

- 1002, Becker, #114 New York, #114 New york , M, 45, 431525, 15000.50
- 1003, Nanapatekar, #115 India, #115 India , M, 45, 431525, 20000.50

6. Delete all the existing rows from Customer table, and let the structure remain itself using TRUNCATE statement.
7. Modify the customer table with the Check constraint to ensure Gender should be either M or F.
8. Display the department from Staff table which has the highest salary by using Inline View.
9. Create a Sequence with the name Seq_Dept on Deptno column of Department_Masters table. It should start from 40 and stop at 200. Increment parameter for the sequence Seq_Dept should be in step of 10.
10. Insert three sample rows by using the above sequence in Department_Masters table.
11. Create a Unique index with the name No_Name on DeptNo and Dname of Department_Masters table.
12. Get information on the index No_Name from the Data Dictionary.

# Lab 5.Data Manipulation Language

| | |
|---|---|
| **Goals** | • Creating table to do the following DML operations<br>• Insert Records<br>• Delete Records<br>• Update Records |
| **Time** | 1 hr |

## 5.1 : Data Manipulation Language

1. Create Employee table with same structure as EMP table.

   SQL>Create table employee as select * from emp where 1=3

   SQL>desc employee

| Name | Null? | Type |
|---|---|---|
| EMPNO | NOT NULL | NUMBER(4) |
| ENAME | | VARCHAR2(10) |
| JOB | | VARCHAR2(50) |
| MGR | | NUMBER(4) |
| HIREDATE | | DATE |
| SAL | | NUMBER(7,2) |
| COMM | | NUMBER(7,2) |
| DEPTNO | | NUMBER(2) |

SQL>select * from employee

2. Write a query to populate Employee table using EMP table's empno, ename, sal, deptno columns.

SQL>select * from employee

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|
| 7369 | SMITH | | | | 800 | | 20 |
| 7499 | ALLEN | | | | 1600 | | 30 |
| 7521 | WARD | | | | 1250 | | 30 |
| 7566 | JONES | | | | 2975 | | 20 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7654 | MARTIN | | | | 1250 | | 30 |
| 7698 | BLAKE | | | | 2850 | | 30 |
| 7782 | CLARK | | | | 2450 | | 10 |
| 7788 | SCOTT | | | | 3000 | | 20 |
| 7839 | KING | | | | 5000 | | 10 |
| 7844 | TURNER | | | | 1500 | | 30 |
| 7876 | ADAMS | | | | 1100 | | 20 |
| 7900 | JAMES | | | | 950 | | 30 |
| 7902 | FORD | | | | 3000 | | 20 |
| 7934 | MILLER | | | | 1300 | | 10 |

14 rows selected.

3. Write a query to change the job and deptno of employee whose empno is 7698 to the job and deptno of employee having empno 7788.
4. Delete the details of department whose department name is ‘SALES’.
5. Write a query to change the deptno of employee with empno 7788 to that of employee having empno 7698.
6. Insert the following rows to the Employee table through parameter substitution.
   - 1000,Allen, Clerk,1001,12-jan-01, 3000, 2,10
   - 1001,George, analyst, null, 08 Sep 92, 5000,0, 10
   - 1002, Becker, Manager, 1000, 4 Nov 92, 2800,4, 20
   - 1003, 'Bill', Clerk, 1002, 4 Nov 92,3000, 0, 20
7. Create a Project Table with below structure

| Name | Null? | Type |
|---|---|---|
| PROJID | NOT NULL | VARCHAR2(10) |
| PROJ_NAME | | VARCHAR2(25) |
| START_DATE | | DATE |
| END_DATE | | DATE |

- Insert Records into Project Table
- Create Employee_Project Table that will have Empno and Project Id as Primary key. Insert records into Employee_Project Table using inline view

# Lab 6.Transaction Control Language Statements

| | |
|---|---|
| **Goals** | • Creating a transaction<br>• Creating a savepoint<br>• Roll back and commit the transaction to the savepoint |
| **Time** | 30 min |

## 6.1 : Transaction Control Language Statements

1. Insert rows with the following data into the Customer table. 6000, John, #115 Chicago, #115 Chicago, M, 25, 7878776, 10000

   - 6001, Jack, #116 France, #116 France, M, 25, 434524, 20000
   - 6002, James, #114 New York, #114 New York, M, 45, 431525, 15000.50

   Use parameter substitution.

2. Create a Savepoint named 'SP1' after third record in the Customer table .

3. Insert the below row in the Customer table.

   6003, John, #114 Chicago, #114 Chicago, M, 45, 439525, 19000.60
4. Execute rollback statement in such a way that whatever manipulations done before Savepoint sp1 are permanently implemented, and the ones after Savepoint SP1 are not stored as a part of the Customer table.

# Appendices

## Appendix A: DBMS SQL Standards

**Key points to keep in mind:**

NA

**How to follow DBMS SQL standards:**

- Decide upon a database naming convention, standardize it across your organization and be consistent in following it. It helps make your code more readable and understandable.
- Tables:
  - Tables represent the instances of an entity. For example, you store all your customer information in a table. Here "customer" is an entity and all the rows in the customers table represent the instances of the entity "customer". So name your table using the entity it represents, namely "Customer". Since the table is storing "multiple instances" of customers, make your table name a plural word.
  - Keeping this in mind:
    - name your customer table as "Customers'"
    - name your order storage table as "Orders"
    - name your error messages table as "ErrorMessages"
  - Suppose your database deals with different logical functions and you want to group your tables according to the logical group they belong to. It will help prefixing your table name with a two or three character prefix that can identify the group.

    For example: Your database has tables which store information about Sales and Human resource departments, then you can name all your tables related to Sales department as shown below:
    - SL_NewLeads

        - SL_Territories
        - SL_TerritoriesManagers
    - You can name all your tables related to Human resources department as shown below:
        - HR_Candidates
        - HR_PremierInstitutes
        - HR_InterviewSchedules
    - Prefix the table names with owner names, as this improves readability, and avoids any unnecessary confusion.
- Primary keys:
    - Primary key is the column(s) that can uniquely identify each row in a table. So just use the column name prefixed with "pk_" + "Table name" for naming primary keys.
    - Given below is an example of how we can name the primary key on the CustomerID column of Customers table:
        - pk_Customers_CustomerID
    - Consider concatenating the column names in case of composite primary keys.
- Foreign keys:
    - Foreign keys are used to represent the relationships between tables which are related. So a foreign key can be considered as a link between the "column of a referencing table" and the "primary key column of the referenced table".
    - We can use the following naming convention for foreign keys:
        - fk_referencing table + referencing column_referenced table + referenced column
    - Based on the above convention, we can name the foreign key, which references the CustomerID column of the Customers table from the Order's tables CustomerID column as shown below:

        - fk_OrdersCustomerID_CustomersCustomerID
    - Foreign key can be composite too. In that case, consider concatenating the column names of referencing and referenced tables while naming the foreign key. This might make the name of the foreign key lengthy. However, you should not be worried about it, as you will never reference this name from your code, except while creating/dropping these constraints.

- Default and Check constraints:
    - Use the column name to which the defaults /check constraints are bound to. Prefix it with “def” and “chk” prefixes respectively for Default and Check constraints.
    - We can name the default constraint for OrderDate Column as def_OrderDate, and the check constraint for OrderDate column as chk_OrderDate.
- Do not use any reserved words for naming my database objects, as that can lead to some unpredictable situations.
- Do not depend on undocumented functionality. The reasons are:
    - you will not get support, when something goes wrong with your undocumented code
    - undocumented functionality is not guaranteed to exist (or behave the same) in a future release or service pack, thereby breaking your code
- Make sure you normalize your data at least till third normal form. At the same time, do not compromise on query performance. A little de-normalization helps queries perform faster.

## Appendix B: Coding Best Practices

Given below are a few best practices in DBMS SQL:

**Do not use SELECT * in your queries. Always write the required column names after the SELECT statement, as shown below:**

SELECT CustomerID, CustomerFirstName, City

This technique results in less disk IO, less network traffic, and hence better performance.

**Try to avoid wildcard characters at the beginning of a word while searching by using the LIKE keyword. This is because this arrangement results in an index scan, which is defeating the purpose of having an index. The following statement results in an index scan, while the second statement results in an index seek:**

1. SELECT LocationID FROM Locations WHERE Specialities LIKE '%pples'
2. SELECT LocationID FROM Locations WHERE Specialities LIKE 'A%s'

Also avoid searching with not equals operators (<> and NOT) as they result in table and index scans. If you must do heavy text-based searches, consider using the Full-Text search feature of SQL Server for better performance.

- Use "Derived tables" wherever possible, as theyperform better. Consider the following query to find the second highest salary from Staff table:

```
SELECT MIN(Salary)
FROM Staff
WHERE EmpID IN
(
SELECT TOP 2 EmpID
FROM Staff
ORDER BY Salary Desc
)
```

The same query can be re-written by using a derived table as shown below, and it performs twice as fast as the above query:

```
SELECT MIN(Salary)
FROM
(
SELECT TOP 2 Salary
FROM Staff
ORDER BY Salary Desc
) AS A
```

This is just an example. The results might differ in different scenarios depending upon the database design, indexes, volume of data, etc. So test all the possible ways a query can be written and go with the efficient one.

- Use char data type for a column, only when the column is non-nullable. If a char column is nullable, it is treated as a fixed length column in SQL Server 7.0+. So a char(100), when NULL, will eat up 100 bytes, resulting in space wastage. So use varchar(100) in this situation. Of course, variable length columns do have a very little processing overhead over fixed length columns. Carefully choose between char and varchar depending upon the length of the data you are going to store.

- Always use a column list in your INSERT statements. This helps in avoiding problems when the table structure changes (like adding a column).

- Do not use the column numbers in the ORDER BY clause as it impairs the readability of the SQL statement. Further, changing the order of columns in the SELECT list has no impact on the ORDER BY when the columns are referred by names instead of numbers. Consider the following example, in which the second query is more readable than the first one:

```
SELECT OrderID, OrderDate
FROM Orders
ORDER BY 2

SELECT OrderID, OrderDate
FROM Orders
ORDER BY OrderDate
```

**Appendix C: Table of Examples**

**Appendix D: Debugging Examples**

1. Identify the mistake in the query (if any) in terms of computed value.
   - SELECT empno, ename, sal+comm as Total FROM emp;

2. CREATE TABLE ITEM_MASTER
   (ITEM_NO VARCHAR2(4) PRIMARY KEY,ITEM_DESC VARCHAR2(25));

```
INSERT INTO ITEM_MASTER VALUES ('1001', 'pen');
INSERT INTO ITEM_MASTER VALUES ('1002', 'Pencil ');
INSERT INTO ITEM_MASTER VALUES ('1003', 'Eracer');
INSERT INTO ITEM_MASTER VALUES ('1104', 'Keyboard MF');
INSERT INTO ITEM_MASTER VALUES ('1005', 'Gel');
INSERT INTO ITEM_MASTER VALUES ('1006', 'chart');
INSERT INTO ITEM_MASTER VALUES ('1007', 'Map');
INSERT INTO ITEM_MASTER VALUES ('1008', 'note book');
INSERT INTO ITEM_MASTER VALUES ('1009', 'STAPLER');
INSERT INTO ITEM_MASTER VALUES ('1010', ' story book');
INSERT INTO ITEM_MASTER VALUES ('1011', 'Calculator');
INSERT INTO ITEM_MASTER VALUES ('2012', 'Writing Pad(S)');
INSERT INTO ITEM_MASTER VALUES ('1013', 'Geometry box');
INSERT INTO ITEM_MASTER VALUES ('1014', 'Id card holder');
INSERT INTO ITEM_MASTER VALUES ('1015', 'FILE');
INSERT INTO ITEM_MASTER VALUES ('1016', 'Bag');
INSERT INTO ITEM_MASTER VALUES ('1017', 'Mobile Pouch');
INSERT INTO ITEM_MASTER VALUES ('1018', 'Punching Machine');
```

a) Correct the mistake in the below query to get the list in ascending order of item_no:
   SELECT item_no, item_desc FROM item_master order by item_no;

b) Identify the mistake, if any to get the list of records correctly:

```
SELECT item_no, item_desc FROM item_master WHERE item_desc like 'k%' or like
'p%' or item_desc like 'S%';
```

Hint: use Conversion Functions for above Qs

3. Rewrite the subquery below to correct the errors(if any)

```
SELECT staff_name,dept_name,staff_sal
      FROM staff_master s,department_master d
       WHERE s.dept_code = sm.dept_code AND --d
          staff_sal < ALL (SELECT AVG(staff_sal) FROM
staff_master sm WHERE department_code = s.dept_code);
```

4. Complete the below query to increase the salary for those people whose salary is less than their Department's average

```
 ---
UPDATE emp a
SET sal = (select avg(sal) FROM enmp b where
....your query....
)
WHERE SAL <
....your query...
```