SUPINFO International University

INSTITUTE OF INFORMATION TECHNOLOGY

Oracle Technologies – SQL 10G

Practice

Oracle – SQL 10g

Objectives

Structured Query Language
Basic Orders
Functions in SQL
Retrieving Data from Several Tables
Advanced Retrieval
Manipulating Data
Creating Schema Objects
Controlling User Access
Managing Schema Objects
Advanced Manipulations
Advanced Group Functions

Version 1.2
Last update: 23/03/2012
Use: SUPINFO

Table des matières

1. Structured Query Language

1.1. Starting with the Virtual Machine

- Name: ORA_WIN_10gR2_v2.4
- Where: courses or \\paris2\LABS\CONTENTS\ORACLE\Virtual Machines
- OS Version: Microsoft Windows XP SP1 VL
- Oracle Version: Oracle Database 10gR1
- OS credentials: oracle/oracle
- Oracle credentials: oracle/oracle

Just download, extract and launch the .vmx file.
No configuration is needed; your Virtual Machine is fully functional.
Once logged in windows, wait while Oracle Databases process initialization.
Next you can connect with SQL*Plus (shortcut on desktop), iSQL*Plus(homepage in Internet Explorer) or SQLDeveloper(in start menu).

2. Basic Orders

2.1. SELECT statement

1. Does *i*SQL*Plus commands access the database?

2. Does the following SELECT statement execute successfully?

```
SELECT last_name, job_id, salary AS Sal
FROM employees;
```

3. Does the following SELECT statement execute successfully?

```
SELECT *
FROM job_grades;
```

4. There are four coding mistakes in this statement:

```
SELECT employee_id, last_name sal x 12 ANNUAL SALARY
FROM employees;
```

You have been hired as SQL programmer for OLCorp. Your first task is to create some reports based on data from the Human Resources tables.

5. You first have to determine the structure of the DEPARTMENTS table and its content.

6. Then, you need to determine the structure of the EMPLOYEES table.

7. The HR department wants a query to display the last name, job code, hire date, and employee number for each employee, with employee number appearing first. Provide an alias STARTDATE for the hire_date column. Save your SQL statement to a file named lab_02_01.sql so that you can disperse this file to the HR department.

8. The HR department needs a query to display all unique job codes from the EMPLOYEES table.

9. The HR department wants more descriptive column headings for its report on employees. Get the statement from lab_02_01.sql in *i*SQL*Plus. Name the column headings Emp #, Employee and Title (case-sensitive).

10. The HR department has requested a report of all employees and their job IDs. Display the last name concatenated with the job ID (separated by comma and space) and name the column “Employee and Title”.

11. To familiarize yourself with the data in the EMPLOYEES table, create a query to display all the data from the EMPLOYEES table. Separate each column output by a comma. Name the column title THE_OUTPUT.

2.2. Restricting and sorting data

1. Due to budget issues, the HR department needs a report that displays the last name and salary of employees earning more than $12,000. Save your statement in lab_02_02.sql.

2. Create a report that displays the last name and department number for employee number 176.

3. The HR departments needs to find high-salary and low-salary employees. Modify lab_02_02.sql to display the last name and salary for all employees whose salary is not in the range of $5,000 to $12,000. Save your statement in lab_02_03.sql.

4. Create a report to display the last name, job id, and start date for the employees with the last names of Matos and Taylor. Order the query in ascending order by start date.

5. The HR department needs your assistance with creating some queries. Display the last name and department number of all employees in department 20 or 50 in ascending alphabetical order by name.

6. Modify lab_02_03.sql to list the last name and salary of employees who earn between $5,000 and $12,000 and are in department 20 or 50. Label the columns “Employee” and “Monthly Salary”, respectively. Resave your script as lab_02_04.sql.

7. The HR department needs a report that displays the last name and hire dates for all employees who were hired id 1994.

8. Create a report to display the last name and job title of all employees who do not have a manager.

9. Display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.

10. Create a report that displays the last name and salary of employees who earn more than an amount that the user specifies prompt. Save the query in lab_02_05.sql.

11. Display all employee last names in which the third letter of the name is “a”.

12. The HR department wants to run reports based on a manager. Create a query that prompts the user for a manager id and generates the employee id, last name, salary, and department for that manager’s employees. The HR department wants the ability to sort the report on a selected column. Test with the following values:
    manager_id = 103, sorted by employees’ last name
    manager_id = 201, sorted by salary
    manager_id = 124, sorted by employee id

13. Display the last name of all employees who have both an “a” and an “e” in their last name.

14. Display the last name, job, and salary for all employees whose job is sales representative (SA_REP) or stock clerk (ST_CLERK) and whose salary is not equal to $2,500, $3,500, or $7,000.

15. Modify lab_02_05.sql to display the last name, salary, and commission for all employees whose commission amount is 20%. Save your statement in lab_02_06.sql.

3. Functions in SQL

3.1. Single Row Functions

1. Write a query to display the current date. Label this column Date.

2. The HR department needs a report to display the employee number, last name, salary, and salary increased by 15.5% (expressed as whole number) for each employee. Label the column "New Salary". Place your SQL statement in a text file named lab_03_01.sql.

3. Modify your previous script to add a column that subtracts the old salary form the new salary. Label the column Increase. Save as lab_03_02.sql.

4. Write a query that displays the last name (with the first letter uppercased and all other letters lowercased) and the length of the last name for all employees whose name starts with the letter J, A, or M. Give each column an appropriate label. Sort by employees' last names.

5. Rewrite the query so that the user is prompted to enter a letter that starts the last name. For example, if the user enters H when prompted for a letter, then the output should show all employees whose last name starts with the letter H.

6. The HR dept' wants to find the length of employment for each employee. Display the last name and calculate the number of months between today and the date on which the employee was hired. Order by the number of months worked. Round up to a whole number.

7. Create a report that produces the following for each employee: <*employee last name*> earns <*salary*> monthly but wants <*3 times salary*>. Label the column "Dream Salaries".

8. Create a query to display the last name and salary for all employees. Format the salary to be 15 characters long, left-padded with $ symbol. Label the column.

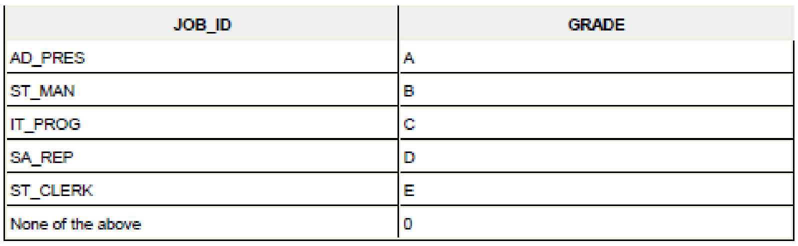
9. Display each employee's last name, hire date, and salary review date, which is the first Monday after six months of service. Label the column REVIEW. Format the dates to appear in the format similar to "Monday, the Thirty-First of July, 2000".

10. Display the last_name, hire date, and day of the week on which the employee started. Label the column DAY. Order the results by the day of the week, starting with Monday.

11. Create a query that displays the employees' last names and commission amounts. If an employee does not earn commission, show "No Commission". Label the column COMM.

12. Create a query that displays the first eight characters of the employees' last names and indicates the amounts of their salaries with asterisks. Each asterisk signifies a thousand dollars. Sort the data in descending order of salary. Label the column EMPLOYEES_AND_THEIR_SALARIES.

13. Using the CASE function, write a query that displays the grade of all employees based on the value of the column JOB_ID, using the following data:



JOB_ID	GRADE
AD_PRES	A
ST_MAN	B
IT_PROG	C
SA_REP	D
ST_CLERK	E
None of the above	0

14. Rewrite the statement in the preceding exercise using the DECODE syntax.

3.2. Group Functions

1. Does a group function work across many rows to produce one result per group?

2. Does a group function include nulls in calculations?

3. Does the WHERE clause restricts rows prior to inclusion in a group calculation?

4. Find the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number. Save your SQL statement in lab_03_03.sql.

5. Modify your script to display the minimum, maximum, sum, and average salary for each job type. Save your script as lab_03_04.sql.

6. Write a query to display the number of people with the same job.

7. Generalize the query so that the user in the HR department is prompted for a job title. Save the script in a file lab_03_05.sql.

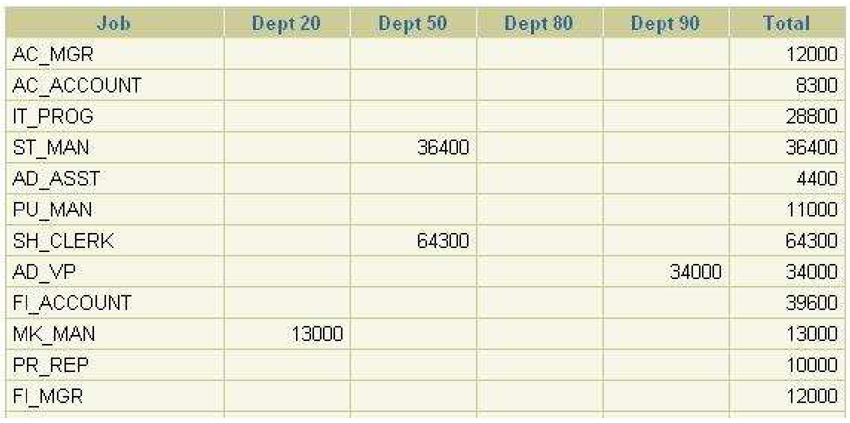
8. Determine the number of managers without listing them. Label the column “Number of Managers”. Use the manager_id column to determine the number of managers.

9. Find the difference between the highest and lowest salaries. Label the column DIFFERENCE.

10. Create a report to display the manager number and the salary of the lowest-paid employee for that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is $6,000 or less. Sort the output in descending order of salary.

11. Create a query that will display the total number of employees and, of that total, the number of employees hired in 1995, 1996, 1997, and 1998. Create appropriate column headings.

12. Create a matrix query to display the job, the salary for that job based on department number, and the total salary for that job, for departments 20, 50, 80, and 90, giving each column an appropriate heading. The required result must correspond to the following table:



Job	Dept 20	Dept 50	Dept 80	Dept 90	Total
AC_MGR					12000
AC_ACCOUNT					8300
IT_PROG					28800
ST_MAN		36400			36400
AD_ASST					4400
PU_MAN					11000
SH_CLERK		64300			64300
AD_VP				34000	34000
FI_ACCOUNT					39600
MK_MAN	13000				13000
PR_REP					10000
FI_MGR					12000

4. Retrieving Data from Several Tables

1. Write a query for the HR department to produce the addresses of all departments. Use the LOCATIONS and COUNTRIES tables. Show the location id, street address, city, state or province, and country in the output. Use a NATURAL JOIN to produce the result.

2. Write a query to display the last name, department number, and department name for all employees.

3. The HR department needs a report of employees in Toronto. Display the last name, job, department number, and department name for all employees who work in Toronto.

4. Create a report to display employees' last name and employee number along with their manager's last name and manager number. Label the columns Employee, Emp#, Manager, and Mgr#, respectively. Place your SQL statement in a text file named lab_04_01.sql.

5. Modify your script to display all employees including King, who has no manager. Order the results by the employee number. Save as lab_04_02.sql.

6. Create a report for the HR department that displays employee last names, department numbers, and all the employees who work in the same department as a given employee. Give each column an appropriate label. Save as lab_04_03.sql.

7. The HR department needs a report on job grades and salaries. To familiarize yourself with the JOB_GRADES table, first show the structure of the JOB_GRADES table. Then create a query that displays the name, job, department name, salary, and grade for all employees.

8. The HR department wants to determine the names of all employees who were hired after Davies. Create a query to display the name and hire date of any employee hired after employee Davies.

9. The HR department needs to find the names and hire dates for all employees who were hired before their managers, along with their managers' names and hire dates. Save the script to a file named lab_04_04.sql.

5. Advanced Retrieval

5.1. Using Subqueries to Solve Queries

1. The HR department needs a query that prompts the user for an employee last name. The query then displays the last name and hire date of any employee in the same department as the employee whose name they supply (excluding that employee). For example, if the user enters Zlotkey, find all employees who work with Zlotkey (excluding Zlotkey).

2. Create a report that displays the employee number, last name, and salary of all employees who earn more than the average salary. Sort the results by ascending salary.

3. Write a query that displays the employee number and last name of all employees who work in a department with any employee whose last name contains a “u”. Place your SQL statement in a text file name lab_05_01.sql.

4. The HR department needs a report that displays the last name, department number, and job id of all employees whose department location id is 1700.

5. Modify the query so that the user is prompted for a location id. Save this to a file named lab_05_02.sql.

6. Create a report for HR that displays the last name and salary of every employee who reports to Steven King.

7. Create a report for HR that displays the department number, last name, and job id for every employee in the Executive department.

8. Modify the query in lab_05_01.sql to display the number, last name, and salary of all employees who earn more than the average salary and who work in a department with any employee whose last name contains an “u”. Save as lab_05_03.sql.

5.2. Using the Set Operators

1. The HR department needs a list of department IDs for departments that do not contain the job id ST_CLERK. Use set operators to create this report.

2. The HR department needs a list of countries that have no departments located in them. Display the country id and the name of the countries. Use set operators to create this report.

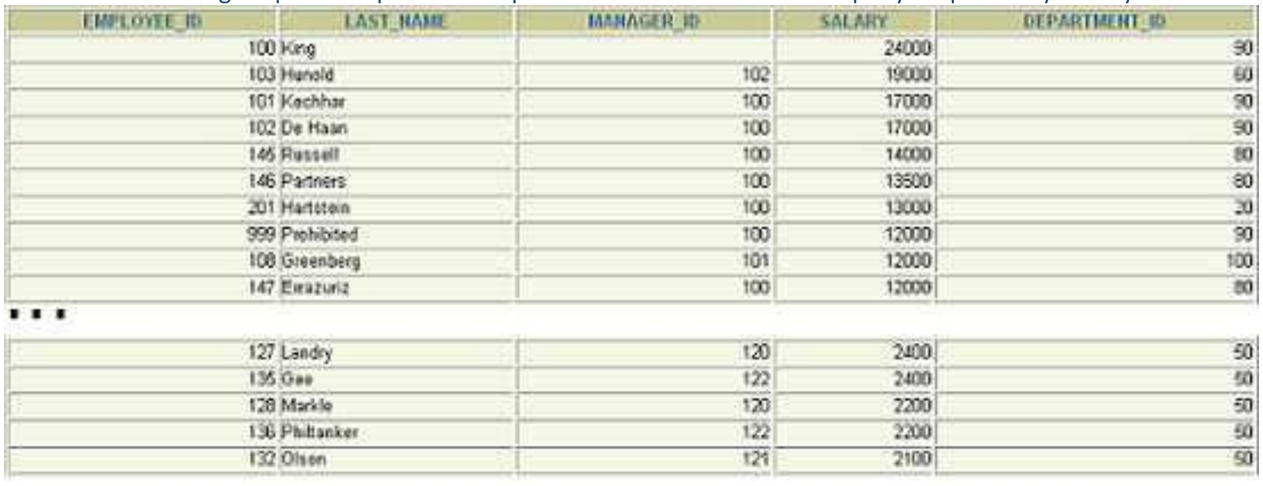
3. Produce a list of jobs for departments 10, 50, and 20, in that order. Display job id and department id using set operators.

4. Create a report that lists the employee IDs and job IDs of those employees who currently have a job title that is the same as their job title when they were initially hired by the company (that is, they changed jobs but have now gone back to doing their original job).

5. The HR department needs a report with the following specifications:
    - Last name and department id of all the employees from the EMPLOYEES table, regardless of whether or not they belong to a department
    - Department id and department name of all the departments from the DEPARTMENTS table, regardless of whether or not they have employees working in them. Write a compound query to accomplish this.

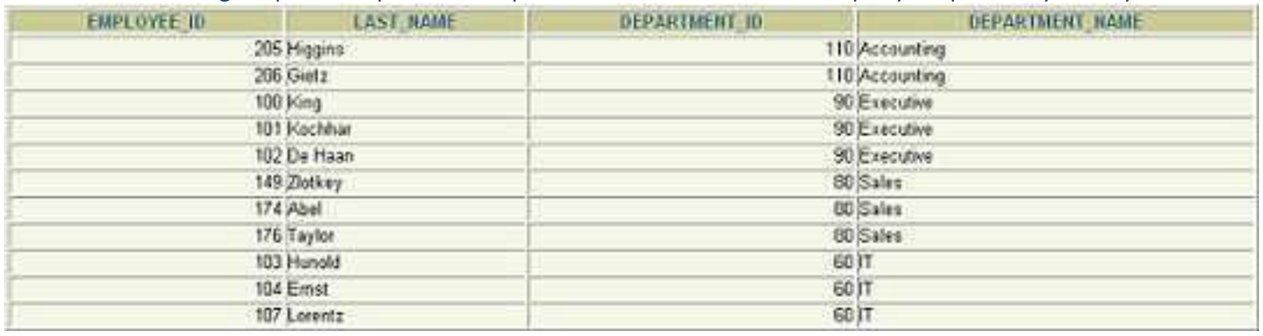
5.3. Hierarchical Retrieval

1. Look at the following output example. Is this input the result of a hierarchical query? Explain why or why not.



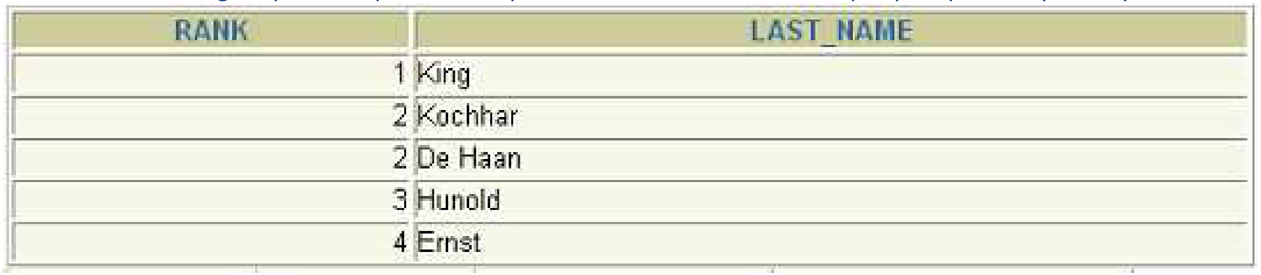
EMPLOYEE_ID	LAST_NAME	MANAGER_ID	SALARY	DEPARTMENT_ID
100	King		24000	90
103	Hunold	102	19000	60
101	Kochhar	100	17000	90
102	De Haan	100	17000	90
145	Russell	100	14000	80
146	Partners	100	13500	80
201	Hartstein	100	13000	20
999	Prohibited	100	12000	90
108	Greenberg	101	12000	100
147	Errazuriz	100	12000	80
...				
127	Landry	120	2400	50
135	Gee	122	2400	50
128	Markle	120	2200	50
136	Philtanker	122	2200	50
132	Olson	121	2100	50

2. Look at the following output example. Is this input the result of a hierarchical query? Explain why or why not.



EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
205	Higgins	110	Accounting
206	Gietz	110	Accounting
100	King	90	Executive
101	Kochhar	90	Executive
102	De Haan	90	Executive
149	Zlotkey	80	Sales
174	Abel	80	Sales
176	Taylor	80	Sales
103	Hunold	60	IT
104	Ernst	60	IT
107	Lorentz	60	IT

3. Look at the following output example. Is this input the result of a hierarchical query? Explain why or why not.



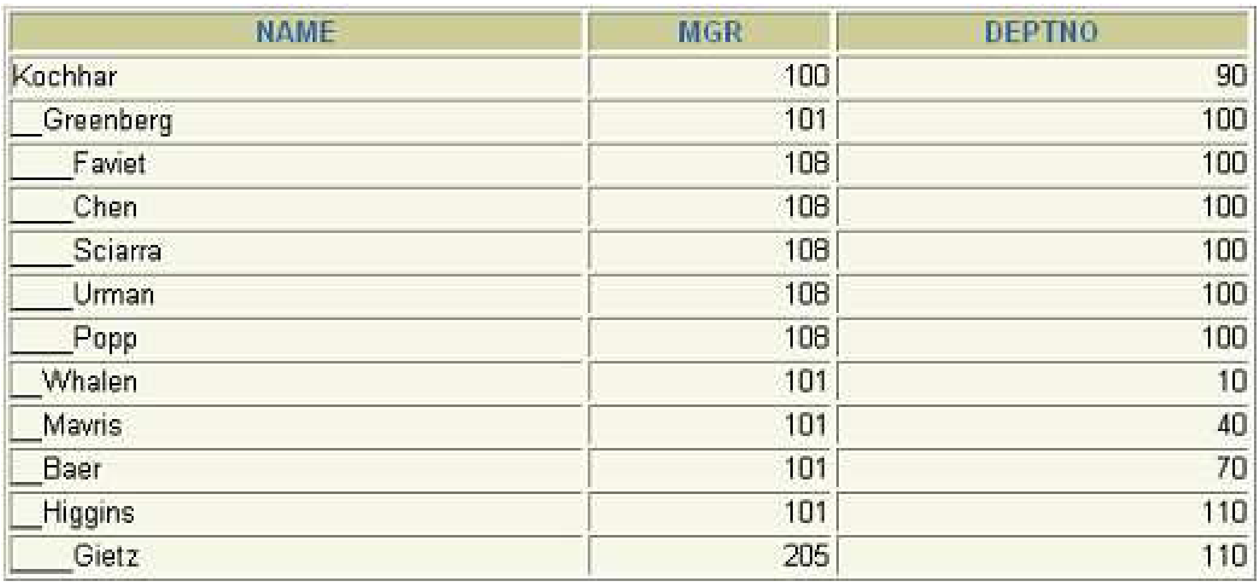
RANK	LAST_NAME
1	King
2	Kochhar
2	De Haan
3	Hunold
4	Ernst

4. Produce a report showing an organization chart for Mourgo's department. Print last names, salaries, and department IDs.

5. Create a report that shows the hierarchy of the managers for the employee Lorentz. Display his immediate manager first.

6. Produce a company organization chart that shows the management hierarchy. Start with the person at the top level, exclude all people with a job id of IT_PROG, and exclude De Haan and those employees who report to De Haan.

7. Create an indented report showing the management hierarchy starting from the employee whose LAST_NAME is Kochhar. Print the employee's last name, manager id, and department id. Give alias names to the columns as shown in the sample output.



NAME	MGR	DEPTNO
Kochhar	100	90
__Greenberg	101	100
____Faviet	108	100
____Chen	108	100
____Sciarra	108	100
____Urman	108	100
____Popp	108	100
__Whalen	101	10
__Mavris	101	40
__Baer	101	70
__Higgins	101	110
____Gietz	205	110

6. Manipulating Data

1. Run the script to create the MY_EMPLOYEE table:

```
CREATE TABLE my_employee
        (id NUMBER(4) CONSTRAINT my_employee_id_nn NOT NULL,
        last_name VARCHAR2(25),
        first_name VARCHAR2(25),
        userid VARCHAR2(8),
        salary NUMBER(9,2));
```

2. Describe the structure of the MY_EMPLOYEE table to identify the column names.

3. Create an INSERT statement to add the first row of data to the MY_EMPLOYEE table from the following sample data. Do not list the column in the INSERT clause.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	spatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	110
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

4. Populate the MY_EMPLOYEE table with the second row of sample data from the preceding list. This time, list the columns explicitly in the INSERT clause.

5. Confirm your addition to the table.

6. Write an insert statement in a dynamic reusable script file named loademp.sql to load rows into the MY_EMPLOYEE table. Concatenate the first letter of the first name and the first seven characters of the last name to produce the user id. Save this script to a file named lab_06_01.sql.
   Then, populate the table with the next two rows of sample data by running the insert statement in the script that you created.

7. Confirm your additions to the table.

8. Make the data additions permanent.

9. Change the last name of employee 3 to Drexler.

10. Change the salary to $1,000 for all employees with a salary less than $900. Verify your changes to the table.

11. Delete Betty Dancs from the MY_EMPLOYEE table and verify your changes.

12. Commit all pending changes.

13. Populate the table with the last row of sample data by using the statements in the script that you created (lab_06_01.sql) and confirm your addition to the table. Then, mark an intermediate point in the processing of the transaction.

14. Empty the entire table (using DML, not DDL!) and confirm that your table is empty.

15. Discard the most recent DELETE operation without discarding the earlier INSERT operation and confirm that the new row is still intact. Make the data addition permanent.

7. Creating Schema Objects

7.1. Creating and Managing Tables

1. Create the DEPT table based on the following specifications:
   - ID, Primary Key, NUMBER, 7 digits
   - NAME, VARCHAR2, 25 characters max.

2. Populate the DEPT table with data from the DEPARTMENTS table, include only columns that you need.

3. Create EMP table based on the following table instance chart. Place the syntax in a script called lab_07_01.sql.

Column name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key type	▨	▨	▨	▨
Nulls/Unique	▨	▨	▨	▨
FK table	▨	▨	▨	DEPT
FK column	▨	▨	▨	ID
Data Type	NUMBER	VARCHAR2	VARCHAR2	NUMBER
Length	7	25	25	7

4. Create the EMP2 table based on the structure of the EMPLOYEES table. Include only the employee_id, first_name, last_name, salary, and department_id columns. Name the columns in the new table id, first_name, last_name, salary and dept_id respectively.

5. Drop the EMP table.

7.2. Creating and Managing Other Schema Objects

1. The staff in the HR department wants to hide some of the data in the EMPLOYEES table. They want a view called EMPLOYEES_VU based on the employee numbers, employee names, and department numbers from the EMPLOYEES table. They want the heading for the employee name to be EMPLOYEE.

2. Confirm that the view works. Display the contents of the EMPLOYEES_VU view.

3. Using your EMPLOYEES_VU view, write a query for the HR department to display all employee names and department numbers.

4. Department 50 needs access to its employee data. Create a view named DEPT50 that contains the employee numbers, employee last names, and department numbers for all employees in department 50. They have requested that you label the view columns EMPNO, EMPLOYEE, and DEPTNO. For security purposes, do not allow an employee to be reassigned to another department through the view.

5. Display the structure and content of the DEPT50 view.

6. Test your view. Attempt to reassign Matos to department 80.

7. You need a sequence that can be used with the primary key column of the DEPT table. The sequence should start at 300 and have a maximum value of 1000. Have your sequence increment by 10. Name the sequence DEPT_ID_SEQ.

8. To test your sequence, write a script to insert two rows in the DEPT table. Name your script lab_07_02.sql. Be sure to use the sequence that you created for the ID column. Add two departments: Education and Administration. Confirm your additions.

9. Create a non-unique index on the NAME column in the DEPT table.

10. Create a synonym for your EMPLOYEES table. Call it EMP.

11. For a specified table, create a script that reports the column names, data types, and data types' lengths, as well as whether nulls are allowed. Prompt the user to enter the table name. Give appropriate aliases to the columns DATA_PRECISION and DATA_SCALE. Save this script in a file named lab_07_03.sql.

12. Create a script that reports the column name, constraint name, constraint type, search condition, and status for a specified table. You must join the USER_CONSTRAINTS and USER_CONS_COLUMNS tables to obtain all of this information. Prompt the user to enter the table name. Save the script in a file named lab_07_04.sql.

13. Add a comment to the DEPARTMENTS table. Then query the USER_TAB_COMMENTS view to verify that the comment is present.

14. Find the names of all synonyms that are in your schema.

15. You need to determine the names and definitions of all of the views in your schema. Create a report that retrieves view information (the view name and text) from the USER_VIEWS data dictionary view. To see more contents of a LONG column, use the SQL*Plus command SET LONG n, where n is the value of the number of characters of the LONG column that you want to see.

16. Find the names of your sequences. Write a query in a script to display the following information about your sequences: sequence name, maximum value, increment size, and last number. Name the script lab_07_05.sql.

8. Controlling User Access

1. What privilege should a user be given to log on to the Oracle server? Is this a system or an object privilege?

2. What privilege should a user be given to create tables?

3. If you create a table, who can pass along privileges to other users on your table?

4. You are the DBA. You are creating many users who require the same system privileges. What should you use to make your job easier?

5. What command do you use to change your password?

6. Create user TOTO and allow him to connect the database. Grant this user read only access to your DEPARTMENTS table.

   Note: You can't give the CREATE SESSION privilege as oracle user on your Virtual Machine. You have to be connected as DBA (B3 curriculum). Enter « CONN / AS SYSDBA »in SQLPlus, give the correct privilege to TOTO and then enter « CONN oracle/oracle ».

7. Query all the rows in your DEPARTMENTS table.

8. Add two new rows to your DEPARTMENTS table:
   - Education as department number 500
   - Human Resources as department number 510

   Connect as TOTO and query your DEPARTMENTS table.

9. Connect as ORACLE. Create a synonym DEPT2 for your DEPARTMENTS table. Then, query all the rows by using your synonym.

10. Query the USER_TABLES data dictionary to see information about the tables that you own. Then query the ALL_TABLES view to see information about all tables you can access.

11. Revoke the SELECT privilege from TOTO.

9. Managing Schema Objects

9.1. Modifying a Table

1. Modify the EMP2 table to allow longer employees’ last names. Confirm your modification.

2. Confirm that both DEPT and EMP2 tables are stored in the data dictionary. (Hint: USER_TABLES)

3. Drop the FIRST_NAME column from the EMP2 table. Confirm your modification by checking the description of the table.

4. In the EMPLOYEES2 table, mark the DEPT_ID column as UNUSED. Confirm your modification by checking the description of the table.

5. Drop all UNUSED columns from the EMP2 table. Confirm your modification.

9.2. Managing Constraints

1. Add a table-level PRIMARY KEY constraint to the EMP2 table on the ID column. The constraint should be named at creation. Name the constraint my_emp_id_pk.

2. Create a PRIMARY KEY constraint to the DEPT table using the ID column. The constraint should be named at creation: my_dept_id_pk.

3. Add a foreign key reference on the EMP2 table that ensures that the employee is not assigned to a nonexistent department. Name the constraint my_emp_dept_id_fk.

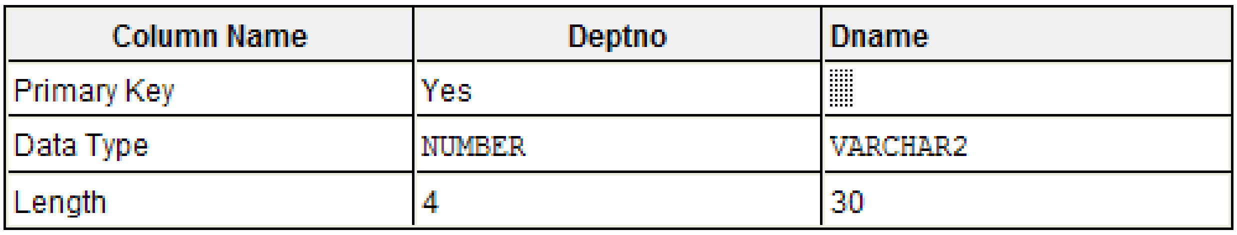
4. Confirm that the constraints were added by querying the USER_CONSTRAINTS view. Note the types and names for the constraints.

5. Display the object names and types from the USER_OBJECTS data dictionary view for the EMP2 and DEPT tables. Notice that the new tables and a new index were created.

6. Modify the EMP2 table. Add a COMMISSION column of NUMBER data type, precision 2, scale 2. Add a constraint to the COMMISSION column that ensures that a commission value is greater than zero.

7. Drop the EMP2 table so that it cannot be restored. Verify the recycle bin.

8. Create the DEPT_NAMES_INDEX table based on the following table instance chart. Name the index for the PRIMARY KEY column as DEPT_PK_IDX.



Column Name	Deptno	Dname
Primary Key	Yes	▨
Data Type	NUMBER	VARCHAR2
Length	4	30

10. Advanced Manipulations

Execute the following script:

```
DROP TABLE sal_history;

CREATE TABLE sal_history
(employee_id NUMBER(6),
hire_date DATE,
salary NUMBER(8,2));

DROP TABLE mgr_history;

CREATE TABLE mgr_history
(employee_id NUMBER(6),
manager_idNUMBER(6),
salary NUMBER(8,2));

DROP TABLE special_sal;

CREATE TABLE special_sal
(employee_id NUMBER(6),
salary NUMBER(8,2));
```

1. Retrieve the details of the employee id, hire date, salary, and manager id of those employees whose employee id is less than 125 from the EMPLOYEES table.
   If the salary is more than $20,000, insert the details of employee id and salary into the SPECIAL_SAL table.
   Else, insert the details of the employee_id, hire date and salary into the SAL_HISTORY table. Insert the details of the employee id, manager id, and salary into the MGR_HISTORY table.

2. Display the records from the three tables (SPECIAL_SAL, SAL_HISTORY, and MGR_HISTORY)

   Execute the following script:

```
DROP TABLE sales_source_data;

CREATE TABLE sales_source_data
(employee_id NUMBER(6),
WEEK_ID NUMBER(2),
SALES_MON NUMBER(8,2),
SALES_TUE NUMBER(8,2),
SALES_WED NUMBER(8,2),
SALES_THUR NUMBER(8,2),
SALES_FRI NUMBER(8,2));

INSERT INTO SALES_SOURCE_DATA VALUES
(178, 6, 1750,2200,1500,1500,3000);
COMMIT;

DROP TABLE sales_info;
CREATE TABLE sales_info
(employee_id NUMBER(6),
WEEK NUMBER(2),
SALES NUMBER(8,2));

INSERT INTO SALES_SOURCE_DATA VALUES
(176, 6, 2000,3000,4000,5000,6000);
COMMIT;
```

3. Write a query to do the following, using a pivoting INSERT statement:
   - Retrieve the details of the employee id, week id, sales on Monday, sales on Tuesday, sales on Wednesday, sales on Thursday, and sales on Friday from the SALES_SOURCE_DATA table.
   - Build a transformation such that each record retrieved from the SALES_SOURCE_DATA table is converted into multiple records for the SALES_INFOS table.

4. Display the records from the SALES_INFO table.

5. You have the data of past employees stored in a flat file called emp.data. You want to store the names and e-mail IDs of all employees past and present in a table. To do this, first create an external table called EMP_DATA using the emp.dat source file in the emp_dir directory. You can use the script in lab_11_01.sql to do this.

   Execute the following script:

```
DROP TABLE emp_hist;

CREATE TABLE emp_hist as
SELECT first_name, last_name, email FROM employees;
```

6. Increase the size of the email column to 45.

7. Merge the data in the EMP_DATA table in the EMP_HIST table. Assume that the data in the external EMP_DATA table is the most up-to-date. If a row in the EMP_DATA table matches the EMP_HIST table, update the email column of the EMP_HIST table to match the EMP_DATA table row. If a row in the EMP_DATA table does not match, insert it into the EMP_HIST table. Rows are considered matching when the employee's first and last names are identical.

8. Retrieve the rows from EMP_HIST after the merge.

   Execute the following script:

```
DROP TABLE emp3;

CREATE TABLE emp3 AS
SELECT * FROM employees;
```

9. In the EMP3 table, change the department for Kochhar to 60 and commit your change. Next, change the department for Kochhar to 50 and commit your change. Track the changes to Kochhar using the Row Version feature.

10. Write a query to display the last name, department number, and salary of any employee whose department number and salary both match the department number and salary of any employee who earns a commission.

11. Display the last name, department name, and salary of any employee whose salary and commission match the salary and commission of any employee located in location ID 1700.

12. Create a query to display the last name, hire date, and salary for all employees who have the same salary and commission as Kochhar. Do not display Kocchar in the result set.

13. Create a query to display the employees who earn a salary that is higher than the salary of all of the sales managers (JOB_ID = 'SA_MAN'). Sort the results on salary from highest to lowest.

14. Display the details of the employee ID, last name, and department ID of those employees who live in cities whose name begins with T.

15. Write a query to find all employees who earn more than the average salary in their departments. Display last name, salary, department ID, and the average salary for the department. Sort by average salary.

16. Find all employees who are not supervisors. Do this by using the NOT EXISTS operator.

17. Rewrite your request using NOT IN operator.

18. Write a query to display the last names of the employees who earn less than the average salary in their departments.

19. Write a query to display the last names of employees who have one or more coworkers in their departments with later hire dates but higher salaries.

20. Write a query to display the employee ID, last names, and department names of all employees. Use a scalar subquery to retrieve the department name in the SELECT statement.

21. Write a query to display the department names of those departments whose total salary cost is above one-eighth (1/8) of the total salary cost of the whole company. Use the WITH clause to write this query. Name the query SUMMARY.

11. Advanced Group Functions

1. Write a query to display the following for those employees whose manager ID is less than 120:
    - Manager ID
    - Job ID and total salary for every job ID for employees who report to the same manager
    - Total salary of those managers
    - Total salary of those managers, irrespective of the job IDs.

2. Observe the output from previous question.
   Write a query using the GROUPING function to determine whether the NULL values in the columns corresponding to the GROUP BY expressions are caused by the ROLLUP Operation.
   Drop all UNUSED columns from the EMP2 table. Confirm your modification.

3. Write a query to display the following for those employees whose manager ID is less than 120:
    - Manager ID
    - Job and total salaries for every job for employees who report to the same manager
    - Total salary of those managers
    - Cross-tabulation values to display the total salary for every job, irrespective of the manager
    - Total salary irrespective of all job titles

4. Observe the output from previous question. Write a query using the GROUPING function to determine whether the NULL values in the columns corresponding to the GROUP BY expressions are caused by the CUBE operation.

5. Using GROUPING SETS, write a query to display the following groupings :
    - department_id, manager_id, job_id
    - department_id, job_id
    - manager_id, job_id
    - The query should calculate the sum of the salaries for each of these groups.