

Oracle Database 10g: SQL Fundamentals II

Student Guide • Volume 2

D17111GC30

Edition 3.0

January 2009

D57874

ORACLE®

Authors

Salome Clement
Chaitanya Koratamaddi
Priya Vennapusa

Technical Contributors and Reviewers

Claire Bennett
Brian Boxx
Zarko Cesljas
Laurent Dereac
Nancy Greenberg
Yash Jain
Angelika Krupp
Malika Marghadi
Priya Nathan
Narayanan Radhakrishnan
Bryan Roberts
Lata Shivaprasad
Naoko Susuki

Editors

Nita Pavitran
Atanu Raychaudhuri

Graphic Designer

Sanjeev Sharma

Publishers

Jobi Varghese
Giri Venugopal

Copyright © 2009, Oracle. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Contents

Preface

I Introduction

- Objectives I-2
- Course Objectives I-3
- Course Overview I-4
- Course Application I-5
- Summary I-6

1 Controlling User Access

- Objectives 1-2
- Controlling User Access 1-3
- Privileges 1-4
- System Privileges 1-5
- Creating Users 1-6
- User System Privileges 1-7
- Granting System Privileges 1-8
- What Is a Role? 1-9
- Creating and Granting Privileges to a Role 1-10
- Changing Your Password 1-11
- Object Privileges 1-12
- Granting Object Privileges 1-14
- Passing On Your Privileges 1-15
- Confirming Privileges Granted 1-16
- Revoking Object Privileges 1-17
- Summary 1-19
- Practice 1: Overview 1-20

2 Managing Schema Objects

- Objectives 2-2
- ALTER TABLE Statement 2-3
- Adding a Column 2-5
- Modifying a Column 2-6
- Dropping a Column 2-7
- SET UNUSED Option 2-8

Adding a Constraint Syntax	2-10
Adding a Constraint	2-11
ON DELETE CASCADE	2-12
Deferring Constraints	2-13
Dropping a Constraint	2-14
Disabling Constraints	2-15
Enabling Constraints	2-16
Cascading Constraints	2-18
Overview of Indexes	2-20
CREATE INDEX with the CREATE TABLE Statement	2-21
Function-Based Indexes	2-23
Removing an Index	2-24
DROP TABLE ... PURGE	2-25
FLASHBACK TABLE Statement	2-26
External Tables	2-28
Creating a Directory for the External Table	2-30
Creating an External Table	2-32
Creating an External Table by Using ORACLE_LOADER	2-34
Querying External Tables	2-36
Summary	2-37
Practice 2: Overview	2-38

3 Manipulating Large Data Sets

Objectives	3-2
Using Subqueries to Manipulate Data	3-3
Copying Rows from Another Table	3-4
Inserting Using a Subquery as a Target	3-5
Retrieving Data with a Subquery as Source	3-7
Updating Two Columns with a Subquery	3-8
Updating Rows Based on Another Table	3-9
Deleting Rows Based on Another Table	3-10
Using the WITH CHECK OPTION Keyword on DML Statements	3-11
Overview of the Explicit Default Feature	3-12
Using Explicit Default Values	3-13
Overview of Multitable INSERT Statements	3-14
Types of Multitable INSERT Statements	3-16
Multitable INSERT Statements	3-17
Unconditional INSERT ALL	3-19
Conditional INSERT ALL	3-20
Conditional INSERT FIRST	3-22

	Pivoting INSERT	3-24
	MERGE Statement	3-27
	MERGE Statement Syntax	3-28
	Merging Rows	3-29
	Tracking Changes in Data	3-31
	Example of the Flashback Version Query	3-32
	VERSIONS BETWEEN Clause	3-34
	Summary	3-35
	Practice 3: Overview	3-36
4	Generating Reports by Grouping Related Data	
	Objectives	4-2
	Review of Group Functions	4-3
	Review of the GROUP BY Clause	4-4
	Review of the HAVING Clause	4-5
	GROUP BY with ROLLUP and CUBE Operators	4-6
	ROLLUP Operator	4-7
	ROLLUP Operator: Example	4-8
	CUBE Operator	4-9
	CUBE Operator: Example	4-10
	GROUPING Function	4-11
	GROUPING Function: Example	4-12
	GROUPING SETS	4-13
	GROUPING SETS: Example	4-15
	Composite Columns	4-17
	Composite Columns: Example	4-19
	Concatenated Groupings	4-21
	Concatenated Groupings: Example	4-22
	Summary	4-23
	Practice 4: Overview	4-24
5	Managing Data in Different Time Zones	
	Objectives	5-2
	Time Zones	5-3
	TIME_ZONE Session Parameter	5-4
	CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP	5-5
	CURRENT_DATE	5-6
	CURRENT_TIMESTAMP	5-7
	LOCALTIMESTAMP	5-8

DBTIMEZONE and SESSIONTIMEZONE 5-9
 TIMESTAMP Data Type 5-10
 TIMESTAMP Data Types 5-11
 TIMESTAMP Fields 5-12
 Difference Between DATE and TIMESTAMP 5-13
 TIMESTAMP WITH TIME ZONE Data Type 5-14
 TIMESTAMP WITH TIMEZONE: Example 5-15
 TIMESTAMP WITH LOCAL TIMEZONE 5-16
 TIMESTAMP WITH LOCAL TIMEZONE: Example 5-17
 INTERVAL Data Types 5-18
 INTERVAL Fields 5-20
 INTERVAL YEAR TO MONTH Data Type 5-21
 INTERVAL YEAR TO MONTH: Example 5-22
 INTERVAL DAY TO SECOND Data Type 5-23
 INTERVAL DAY TO SECOND Data Type: Example 5-24
 EXTRACT 5-25
 TZ_OFFSET 5-26
 TIMESTAMP Conversion Using FROM_TZ 5-28
 Converting to TIMESTAMP Using TO_TIMESTAMP and TO_TIMESTAMP_TZ 5-29
 Time Interval Conversion with TO_YMINTERVAL 5-30
 Using TO_DSINTERVAL: Example 5-31
 Daylight Saving Time 5-32
 Summary 5-34
 Practice 5: Overview 5-35

6 Retrieving Data Using Subqueries

Objectives 6-2
 Multiple-Column Subqueries 6-3
 Column Comparisons 6-4
 Pairwise Comparison Subquery 6-5
 Nonpairwise Comparison Subquery 6-6
 Scalar Subquery Expressions 6-7
 Scalar Subqueries: Examples 6-8
 Correlated Subqueries 6-10
 Using Correlated Subqueries 6-12
 Using the EXISTS Operator 6-14
 Find Employees Who Have At Least One Person Reporting to Them 6-15
 Find All Departments That Do Not Have Any Employees 6-16
 Correlated UPDATE 6-17

Using Correlated `UPDATE` 6-18

Correlated `DELETE` 6-20

Using Correlated `DELETE` 6-21

`WITH` Clause 6-22

`WITH` Clause: Example 6-23

Summary 6-25

Practice 6: Overview 6-27

7 Hierarchical Retrieval

Objectives 7-2

Sample Data from the `EMPLOYEES` Table 7-3

Natural Tree Structure 7-4

Hierarchical Queries 7-5

Walking the Tree 7-6

Walking the Tree: From the Bottom Up 7-8

Walking the Tree: From the Top Down 7-9

Ranking Rows with the `LEVEL` Pseudocolumn 7-10

Formatting Hierarchical Reports Using `LEVEL` and `LPAD` 7-11

Pruning Branches 7-13

Summary 7-14

Practice 7: Overview 7-15

8 Regular Expression Support

Objectives 8-2

Regular Expression: Overview 8-3

Meta Characters 8-4

Using Meta Characters 8-5

Regular Expression Functions 8-7

`REGEXP` Function Syntax 8-8

Performing Basic Searches 8-9

Checking the Presence of a Pattern 8-10

Example of Extracting Substrings 8-11

Replacing Patterns 8-12

Regular Expressions and Check Constraints 8-13

Summary 8-14

Practice 8: Overview 8-15

Appendix A: Practice Solutions

Appendix B: Table Descriptions and Data

Appendix C: Writing Advanced Scripts

- Objectives C-2
- Using SQL to Generate SQL C-3
- Creating a Basic Script C-4
- Controlling the Environment C-5
- The Complete Picture C-6
- Dumping the Contents of a Table to a File C-7
- Generating a Dynamic Predicate C-9
- Summary C-11

Appendix D: Oracle Architectural Components

- Objectives D-2
- Oracle Database Architecture: Overview D-3
- Database Physical Architecture D-4
- Control Files D-5
- Redo Log Files D-6
- Tablespaces and Data Files D-7
- Segments, Extents, and Blocks D-8
- Oracle Instance Management D-9
- Oracle Memory Structures D-10
- Oracle Processes D-12
- Other Key Physical Structures D-13
- Processing a SQL Statement D-14
- Connecting to an Instance D-15
- Processing a Query D-17
- Shared Pool D-18
- Database Buffer Cache D-20
- Program Global Area (PGA) D-21
- Processing a DML Statement D-22
- Redo Log Buffer D-24
- Rollback Segment D-25
- COMMIT Processing D-26
- Summary D-28

Appendix E: Using SQL Developer

- Objectives E-2
- What Is Oracle SQL Developer? E-3
- Key Features E-4
- Installing SQL Developer E-5
- Menus for SQL Developer E-6
- Creating a Database Connection E-7

Browsing Database Objects	E-9
Creating a Schema Object	E-10
Creating a New Table: Example	E-11
Using SQL Worksheet	E-12
Executing SQL Statements	E-14
Viewing the Execution Plan	E-15
Formatting the SQL Code	E-16
Using Snippets	E-17
Using Snippets: Example	E-18
Using SQL*Plus	E-19
Database Reporting	E-20
Creating a User Defined Report	E-21
Summary	E-22

Index

Additional Practices

Additional Practice Solutions

Additional Practices

sharat b (bsaratchand@gmail.com) has a non-transferable license to use this Student Guide.

Additional Practices

The following exercises can be used for extra practice after you have discussed data manipulation language (DML) and data definition language (DDL) statements in the lessons titled “Managing Schema Objects” and “Manipulating Large Data Sets.”

Note: Run the `lab_ap_cre_special_sal.sql`, `lab_ap_cre_sal_history.sql`, and `lab_ap_cre_mgr_history.sql` scripts in the labs folder to create the `SPECIAL_SAL`, `SAL_HISTORY`, and `MGR_HISTORY` tables.

1. The Human Resources department wants to get a list of underpaid employees, the salary history of employees, and the salary history of managers based on an industry salary survey. So they have asked you to do the following:

Write a statement to do the following:

- Retrieve the employee ID, hire date, salary, and manager ID of those employees whose employee ID is more than or equal to 200 from the `EMPLOYEES` table.
- If the salary is less than \$5,000, insert the employee ID and salary into the `SPECIAL_SAL` table.
- Insert the employee ID, hire date, and salary into the `SAL_HISTORY` table.
- Insert the employee ID, manager ID, and salary into the `MGR_HISTORY` table.

2. Query the `SPECIAL_SAL`, `SAL_HISTORY` and `MGR_HISTORY` tables to view the inserted records.

`SPECIAL_SAL`

	EMPLOYEE_ID	SALARY
1	200	4400

`SALARY_HISTORY`

	EMPLOYEE_ID	HIRE_DATE	SALARY
1	201	17-FEB-1996	13000
2	202	17-AUG-1997	6000
3	203	07-JUN-1994	6500
4	204	07-JUN-1994	10000
5	205	07-JUN-1994	12000
6	206	07-JUN-1994	8300

Additional Practices (continued)

MGR_HISTORY

	EMPLOYEE_ID	MANAGER_ID	SALARY
1	201	100	13000
2	202	201	6000
3	203	101	6500
4	204	101	10000
5	205	101	12000
6	206	205	8300

3. The DBA wants you to create a table, which has a primary key constraint, but wants the index to have a different name than the constraint. Create the LOCATIONS_NAMED_INDEX table based on the following table instance chart. Name the index for the PRIMARY KEY column as LOCATIONS_PK_IDX.

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

4. Query the USER_INDEXES table to display INDEX_NAME for the LOCATIONS_NAMED_INDEX table.

	INDEX_NAME	TABLE_NAME
1	LOCATIONS_PK_IDX	LOCATIONS_NAMED_INDEX

Additional Practices (continued)

The following exercises can be used for extra practice after you have discussed enhancements to the GROUP BY clause.

5. The Human Resources department requires some reports on certain departments. These are its requirements:

Write a query to display the following for those departments whose department ID is greater than 80:

- The total salary for every job within a department
- The total salary
- The total salary for those cities in which the departments are located
- The total salary for every job, irrespective of the department
- The total salary for every department irrespective of the city
- The total salary for the departments, irrespective of the job titles and cities

R	CITY	R	DEPARTMENT_NAME	R	JOB_ID	R	SUM(E.SALARY)
1	(null)	(null)	(null)	(null)			129900
2	(null)	(null)		AD_VP			34000
3	(null)	(null)		AC_MGR			12000
4	(null)	(null)		FI_MGR			12000
5	(null)	(null)		AD_PRES			24000
6	(null)	(null)		AC_ACCOUNT			8300
7	(null)	(null)		FI_ACCOUNT			39600
8	(null)	Finance		(null)			51600
9	(null)	Finance		FI_MGR			12000
10	(null)	Finance		FI_ACCOUNT			39600
11	(null)	Executive		(null)			58000
12	(null)	Executive		AD_VP			34000
13	(null)	Executive		AD_PRES			24000
14	(null)	Accounting		(null)			20300
15	(null)	Accounting		AC_MGR			12000
16	(null)	Accounting		AC_ACCOUNT			8300
17	Seattle	(null)		(null)			129900
18	Seattle	(null)		AD_VP			34000
19	Seattle	(null)		AC_MGR			12000
20	Seattle	(null)		FI_MGR			12000
21	Seattle	(null)		AD_PRES			24000
22	Seattle	(null)		AC_ACCOUNT			8300
23	Seattle	(null)		FI_ACCOUNT			39600
24	Seattle	Finance		(null)			51600
25	Seattle	Finance		FI_MGR			12000
26	Seattle	Finance		FI_ACCOUNT			39600
27	Seattle	Executive		(null)			58000
28	Seattle	Executive		AD_VP			34000
29	Seattle	Executive		AD_PRES			24000
30	Seattle	Accounting		(null)			20300
31	Seattle	Accounting		AC_MGR			12000
32	Seattle	Accounting		AC_ACCOUNT			8300

Additional Practices (continued)

6. The Accounting department requires an analysis on the maximum and minimum salaries by department, job, and manager. They have asked you to do the following:

Write a query to display the following groupings:

- Department ID, Job ID
- Job ID, Manager ID

The query should calculate the maximum and minimum salaries for each of these groups.

	DEPARTMENT_ID	JOB_ID	MANAGER_ID	MAX(SALARY)	MIN(SALARY)
1	(null)	AC_MGR	101	12000	12000
2	(null)	SH_CLERK	122	3800	2500
3	(null)	SH_CLERK	124	3100	2600
4	(null)	MK_MAN	100	13000	13000
5	(null)	ST_MAN	100	8200	5800
6	(null)	ST_CLERK	121	3300	2100
7	(null)	SA_REP	148	11500	6100
8	(null)	SH_CLERK	120	3200	2500
9	(null)	AD_ASST	101	4400	4400
10	(null)	AD_PRES	(null)	24000	24000

...

40	50	SH_CLERK	(null)	4200	2500
41	20	MK_MAN	(null)	13000	13000
42	90	AD_PRES	(null)	24000	24000
43	60	IT_PROG	(null)	9000	4200
44	100	FI_MGR	(null)	12000	12000
45	30	PU_CLERK	(null)	3100	2500
46	100	FI_ACCOUNT	(null)	9000	6900
47	70	PR_REP	(null)	10000	10000
48	(null)	SA_REP	(null)	7000	7000
49	10	AD_ASST	(null)	4400	4400
50	20	MK_REP	(null)	6000	6000
51	40	HR_REP	(null)	6500	6500
52	30	PU_MAN	(null)	11000	11000

Additional Practices (continued)

The following exercises can be used for extra practice after you have discussed the datetime functions.

You work for a global company and the new vice president of operations wants to know the different time zones of all the company branches. He has requested the following information:

7. Alter the session to set the NLS_DATE_FORMAT to DD-MON-YYYY HH24:MI:SS.
8. a. Write queries to display the time zone offsets (TZ_OFFSET) for the following time zones:

Australia/Sydney

	TZ_OFFSET('AUSTRALIA/SYDNEY')
1	+11:00

Chile/Easter Island

	TZ_OFFSET('CHILE/EASTERISLAND')
1	-05:00

- b. Alter the session to set the TIME_ZONE parameter value to the time zone offset of Australia/Sydney.
- c. Display SYSDATE, CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP for this session.

Note: The output may be different based on the date when the command is executed.

	SYSDATE	CURRENT_DATE	CURRENT_TIMESTAMP	LOCALTIMESTAMP
1	28-NOV-2008 05:10:32	28-NOV-2008 20:10:32	28-NOV-08 08.10.32.230524000 PM +10:00	28-NOV-08 08.10.32.230524000 PM

- d. Alter the session to set the TIME_ZONE parameter value to the time zone offset of Chile/Easter Island.

Note: The results of the preceding question are based on a different date, and in some cases, they will not match the actual results that the students get. Also, the time zone offset of the various countries may differ based on the daylight saving time.

- e. Display SYSDATE, CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP for this session.

Note: The output may be different based on the date when the command is executed.

	SYSDATE	CURRENT_DATE	CURRENT_TIMESTAMP	LOCALTIMESTAMP
1	28-NOV-2008 05:13:19	28-NOV-2008 04:13:19	28-NOV-08 04.13.19.415937000 AM -06:00	28-NOV-08 04.13.19.415937000 AM

- f. Alter the session to set the NLS_DATE_FORMAT to DD-MON-YYYY.

Additional Practices (continued)

Note

- Observe in the preceding question that CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP are all sensitive to the session time zone. Observe that SYSDATE is not sensitive to the session time zone.
- The results of the preceding question are based on a different date, and in some cases, they will not match the actual results that the students get. Also the time zone offset of the various countries may differ based on the daylight saving time.

9. The Human Resources department wants a list of employees who are up for review in January, so they have requested you to do the following:

Write a query to display the last name, month of the date of hire, and hire date of those employees who have been hired in the month of January, irrespective of the year of hire.

	LAST_NAME	EXTRACT(MONTHFROMHIRE_DATE)	HIRE_DATE
1	Grant		13-JAN-2000
2	De Haan		13-JAN-1993
3	Hunold		03-JAN-1990
4	Landry		14-JAN-1999
5	Davies		29-JAN-1997
6	Partners		05-JAN-1997
7	Zlotkey		29-JAN-2000
8	Tucker		30-JAN-1997
9	King		30-JAN-1996
10	Marvins		24-JAN-2000
11	Fox		24-JAN-1998
12	Johnson		04-JAN-2000
13	Taylor		24-JAN-1998
14	Sarchand		27-JAN-1996

Additional Practices (continued)

The following exercises can be used for extra practice after you have discussed advanced subqueries.



10. The CEO needs a report on the top three earners in the company for profit sharing. He has asked you to provide him with a list.

Write a query to display the top three earners in the EMPLOYEES table. Display their last names and salaries.

	 LAST_NAME	 SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000

11. The benefits for the state of California have been changed based on a local ordinance. So the benefits representative has asked you to compile a list of people who are affected. Write a query to display the employee ID and last name of the employees who work in the state of California.

Hint: Use scalar subqueries.

	 EMPLOYEE_ID	 LAST_NAME
1	198	OConnell
2	199	Grant
3	120	Weiss
4	121	Fripp
5	122	Kaufling
6	123	Vollman
7	124	Mourgos
8	125	Nayer

• • •

41	193	Everett
42	194	McCain
43	195	Jones
44	196	Walsh
45	197	Feeney

12. The DBA wants to remove old information from the database. One of the things that the DBA thinks is unnecessary is the old employment records. She has asked you to do the following:

Write a query to delete the oldest JOB_HISTORY row of an employee by looking up the JOB_HISTORY table for MIN(START_DATE) for the employee. Delete the records of *only* those employees who have changed at least two jobs.

Hint: Use a correlated DELETE command.

Additional Practices (continued)

13. The vice president of Human Resources needs the complete employment records for his annual employee recognition banquet speech. He makes a quick phone call to stop you from following the DBA's orders.

Roll back the transaction.

14. The sluggish economy is forcing the management to take cost reduction actions. The CEO wants to review the highest paid jobs in the company. He has requested a list from you based on the following specifications:

Write a query to display the job IDs of those jobs whose maximum salary is above half the maximum salary in the entire company. Use the WITH clause to write this query. Name the query MAX_SAL_CALC.

	JOB_TITLE	JOB_TOTAL
1	President	24000
2	Administration Vice President	17000
3	Sales Manager	14000
4	Marketing Manager	13000

The following exercises can be used for extra practice after you have discussed hierarchical retrieval.

15. Lex De Haan is quitting the company. His replacement wants reports of his direct reports.

Write a SQL statement to display the employee number, last name, start date, and salary, showing:

- a. De Haan's direct reports:

	EMPLOYEE_ID	LAST_NAME	HIRE_DATE	SALARY
1	103	Hunold	03-JAN-1990	9000

- b. The organization tree under De Haan (employee number 102):

	EMPLOYEE_ID	LAST_NAME	HIRE_DATE	SALARY
1	103	Hunold	03-JAN-1990	9000
2	104	Ernst	21-MAY-1991	6000
3	105	Austin	25-JUN-1997	4800
4	106	Pataballa	05-FEB-1998	4800
5	107	Lorentz	07-FEB-1999	4200

16. Write a hierarchical query to display the employee number, manager number, and employee last name for all employees who are two levels below employee De Haan (employee number 102). Also, display the level of the employee.

	EMPLOYEE_ID	MANAGER_ID	LEVEL	LAST_NAME
1	104	103	3	Ernst
2	105	103	3	Austin
3	106	103	3	Pataballa
4	107	103	3	Lorentz

Additional Practices (continued)

17. The CEO wants a hierarchical report on all employees. He has given you the following requirements:

Produce a hierarchical report to display the employee number, manager number, the LEVEL pseudocolumn, and employee last name. For every row in the EMPLOYEES table, you should print a tree structure that shows the employee, the employee's manager, the manager's manager, and so on. Use indentations for the NAME column.

	EMPLOYEE_ID	MANAGER_ID	LEVEL	LAST_NAME
1	100	(null)	1	King
2	101	100	1	Kochhar
3	100	(null)	2	__King
4	102	100	1	De Haan
5	100	(null)	2	__King
6	103	102	1	Hunold
7	102	100	2	__De Haan
8	100	(null)	3	___King

...

310	101	100	2	__Kochhar
311	100	(null)	3	___King
312	206	205	1	Gietz
313	205	101	2	__Higgins
314	101	100	3	___Kochhar
315	100	(null)	4	____King

Note: The output shown is only a sample. All the rows from the actual output are not included here.

Additional Practice Solutions

sharat b (bsaratchand@gmail.com) has a non-transferable license to use this Student Guide.

Additional Practices Solutions

The following exercises can be used for extra practice after you have discussed the data manipulation language (DML) and data definition language (DDL) statements in the lessons titled “Managing Schema Objects” and “Manipulating Large Data Sets.”

Note: Run the `lab_ap_cre_special_sal.sql`, `lab_ap_cre_sal_history.sql`, and `lab_ap_cre_mgr_history.sql` scripts in the labs folder to create the `SPECIAL_SAL`, `SAL_HISTORY`, and `MGR_HISTORY` tables.

1. The Human Resources department wants a list of underpaid employees, the salary history of employees, and the salary history of managers based on an industry salary survey. So they have asked you to do the following:

Write a statement to do the following:

- Retrieve the employee ID, hire date, salary, and manager ID of those employees whose employee ID is more than or equal to 200 from the `EMPLOYEES` table.
- If the salary is less than \$5,000, insert the employee ID and salary into the `SPECIAL_SAL` table.
- Insert the employee ID, hire date, and salary into the `SAL_HISTORY` table.
- Insert the employee ID, manager ID, and salary into the `MGR_HISTORY` table.

```
INSERT ALL
WHEN SAL < 5000 THEN
  INTO special_sal VALUES (EMPID, SAL)
ELSE
  INTO sal_history VALUES (EMPID, HIREDATE, SAL)
  INTO mgr_history VALUES (EMPID, MGR, SAL)
SELECT employee_id EMPID, hire_date HIREDATE,
       salary SAL, manager_id MGR
FROM employees
WHERE employee_id >= 200;
```

2. Query the `SPECIAL_SAL`, `SAL_HISTORY`, and `MGR_HISTORY` tables to view the inserted records.

```
SELECT * FROM special_sal;
SELECT * FROM sal_history;
SELECT * FROM mgr_history;
```

Additional Practices Solutions (continued)

3. The DBA wants you to create a table, which has a primary key constraint, but the DBA wants the index to have a different name than the constraint. Create the `LOCATIONS_NAMED_INDEX` table based on the following table instance chart.

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

Name the index for the PRIMARY KEY column as `LOCATIONS_PK_IDX`.

```
CREATE TABLE LOCATIONS_NAMED_INDEX
(location_id NUMBER(4) PRIMARY KEY USING INDEX
(CREATE INDEX locations_pk_idx ON
LOCATIONS_NAMED_INDEX(location_id)),
location_name VARCHAR2(20));
```

4. Query the `USER_INDEXES` table to display `INDEX_NAME` for the `LOCATIONS_NAMED_INDEX` table.

```
SELECT INDEX_NAME, TABLE_NAME
FROM USER_INDEXES
WHERE TABLE_NAME = 'LOCATIONS_NAMED_INDEX';
```

The following exercises can be used for extra practice after you have discussed enhancements to the `GROUP BY` clause.

5. The Human Resources department requires some reports on certain departments. These are its requirements:

Write a query to display the following for those departments whose department ID is greater than 80:

- The total salary for every job within a department
- The total salary
- The total salary for those cities in which the departments are located
- The total salary for every job, irrespective of the department
- The total salary for every department irrespective of the city
- The total salary for the departments, irrespective of the job titles and cities

Additional Practices Solutions (continued)

```
COLUMN      city FORMAT A25 Heading CITY
COLUMN      department_name FORMAT A15 Heading DNAME
COLUMN      job_id FORMAT A10 Heading JOB
COLUMN      SUM(salary) FORMAT $99,99,999.00 Heading
            SUM(SALARY)

SELECT      l.city,d.department_name, e.job_id,
            SUM(e.salary)
FROM        locations l, employees e, departments d
WHERE       d.location_id = l.location_id
AND         e.department_id = d.department_id
AND         e.department_id > 80
GROUP BY CUBE( l.city, d.department_name, e.job_id);
```

6. The Accounting department requires an analysis on the maximum and minimum salaries by department, job, and manager. They have asked you to do the following:

Write a query to display the following groupings:

- Department ID, Job ID
- Job ID, Manager ID

The query should calculate the maximum and minimum salaries for each of these groups.

```
SELECT
    department_id,job_id,manager_id,max(salary),
    min(salary)
FROM    employees
GROUP BY GROUPING SETS
        ((department_id,job_id), (job_id,manager_id));
```

The following exercises can be used for extra practice after you have discussed the datetime functions.

You work for a global company and the new vice president of operations wants to know the different time zones of all the company branches. He has requested the following information:

7. Alter the session to set the NLS_DATE_FORMAT to DD-MON-YYYY HH24:MI:SS.

```
ALTER SESSION
SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
```


Additional Practices Solutions (continued)

8. a. Write queries to display the time zone offsets (TZ_OFFSET) for the following time zones:
- Australia/Sydney

```
SELECT TZ_OFFSET ('Australia/Sydney') from dual;
```

- Chile/Easter Island

```
SELECT TZ_OFFSET ('Chile/EasterIsland') from dual;
```

- b. Alter the session to set the TIME_ZONE parameter value to the time zone offset of Australia/Sydney.

```
ALTER SESSION SET TIME_ZONE = '+10:00';
```

- c. Display SYSDATE, CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP for this session.

Note: The output may be different based on the date when the command is executed.

```
SELECT SYSDATE, CURRENT_DATE, CURRENT_TIMESTAMP,  
LOCALTIMESTAMP FROM DUAL;
```

- d. Alter the session to set the TIME_ZONE parameter value to the time zone offset of Chile/Easter Island.

Note: The results of the preceding question are based on a different date, and in some cases, they will not match the actual results that the students get. Also, the time zone offset of the various countries may differ based on the daylight saving time.

```
ALTER SESSION SET TIME_ZONE = '-06:00';
```

- e. Display SYSDATE, CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP for this session.

Note: The output may be different based on the date when the command is executed.

```
SELECT SYSDATE, CURRENT_DATE, CURRENT_TIMESTAMP,  
LOCALTIMESTAMP FROM DUAL;
```

- f. Alter the session to set NLS_DATE_FORMAT to DD-MON-YYYY.

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY';
```

Additional Practices Solutions (continued)

Note

- Observe in the preceding question that CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP are all sensitive to the session time zone. Observe that SYSDATE is not sensitive to the session time zone.
- The results of the preceding question are based on a different date, and in some cases, they will not match the actual results that the students get. Also, the time zone offset of the various countries may differ based on the daylight saving time.

9. The Human Resources department wants a list of employees who are up for review in January, so they have requested you to do the following:

Write a query to display the last names, month of the date of hire, and hire date of those employees who have been hired in the month of January, irrespective of the year of hire.

```
SELECT last_name, EXTRACT (MONTH FROM HIRE_DATE),  
       HIRE_DATE FROM employees  
WHERE EXTRACT (MONTH FROM HIRE_DATE) = 1;
```

The following exercises can be used for extra practice after you have discussed advanced subqueries.

10. The CEO needs a report on the top three earners in the company for profit sharing. He has asked you to provide him with a list.

Write a query to display the top three earners in the EMPLOYEES table. Display their last names and salaries.

```
SELECT last_name, salary  
FROM employees e  
WHERE 3 > (SELECT COUNT (*)  
FROM employees  
WHERE e.salary < salary);
```

11. The benefits for the state of California have been changed based on a local ordinance. So the benefits representative has asked you to compile a list of people who are affected.

Write a query to display the employee ID and last name of the employees who work in the state of California.

Hint: Use scalar subqueries.

Additional Practices Solutions (continued)

```
SELECT employee_id, last_name
FROM employees e
WHERE ((SELECT location_id
        FROM departments d
        WHERE e.department_id = d.department_id)
        IN (SELECT location_id
            FROM locations l
            WHERE state_province = 'California'));
```

12. The DBA wants to remove old information from the database. One of the things that the DBA thinks is unnecessary is the old employment records. He or she has asked you to do the following:

Write a query to delete the oldest JOB_HISTORY row of an employee by looking up the JOB_HISTORY table for MIN(START_DATE) for the employee. Delete the records of *only* those employees who have changed at least two jobs.

Hint: Use a correlated DELETE command.

```
DELETE FROM job_history JH
WHERE employee_id = (SELECT employee_id
                    FROM employees E
                    WHERE JH.employee_id = E.employee_id
                    AND START_DATE = (SELECT MIN(start_date)
                                      FROM job_history JH
                                      WHERE JH.employee_id = E.employee_id)
                    AND 3 > (SELECT COUNT(*)
                              FROM job_history JH
                              WHERE JH.employee_id = E.employee_id
                              GROUP BY EMPLOYEE_ID
                              HAVING COUNT(*) >= 2));
```

13. The vice president of Human Resources needs the complete employment records for his annual employee recognition banquet speech. He makes a quick phone call to stop you from following the DBA's orders.

Roll back the transaction.

```
ROLLBACK;
```

Additional Practices Solutions (continued)

14. The sluggish economy is forcing the management to take cost reduction actions. The CEO wants to review the highest paid jobs in the company. He has requested a list from you based on the following specifications:

Write a query to display the job IDs of those jobs whose maximum salary is above half the maximum salary in the entire company. Use the WITH clause to write this query. Name the query MAX_SAL_CALC.

```
WITH
MAX_SAL_CALC AS (SELECT job_title, MAX(salary) AS
job_total
FROM employees, jobs
WHERE employees.job_id = jobs.job_id
GROUP BY job_title)
SELECT job_title, job_total
FROM MAX_SAL_CALC
WHERE job_total > (SELECT MAX(job_total) * 1/2
FROM MAX_SAL_CALC)
ORDER BY job_total DESC;
```

The following exercises can be used for extra practice after you have discussed hierarchical retrieval.

15. Lex De Haan is quitting the company. His replacement wants reports of his direct reports.

Write a SQL statement to display the employee number, last name, start date, and salary, showing:

- a. De Haan's direct reports:

```
SELECT employee_id, last_name, hire_date, salary
FROM employees
WHERE manager_id = (SELECT employee_id
FROM employees
WHERE last_name = 'De Haan');
```

- b. The organization tree under De Haan (employee number 102):

```
SELECT employee_id, last_name, hire_date, salary
FROM employees
WHERE employee_id != 102
CONNECT BY manager_id = PRIOR employee_id
START WITH employee_id = 102;
```

Additional Practices Solutions (continued)

16. Write a hierarchical query to display the employee number, manager number, and employee last name for all employees who are two levels below employee De Haan (employee number 102). Also display the level of the employee.

```
SELECT employee_id,manager_id,level,last_name
FROM employees
WHERE LEVEL = 3
CONNECT BY manager_id = PRIOR employee_id
START WITH employee_id = 102;
```

17. The CEO wants a hierarchical report on all employees. He has given you the following requirements:

Produce a hierarchical report to display the employee number, manager number, the LEVEL pseudocolumn, and employee last name. For every row in the EMPLOYEES table, you should print a tree structure that shows the employee, the employee's manager, the manager's manager, and so on. Use indentations for the NAME column.

```
COLUMN name FORMAT A25
SELECT employee_id,manager_id,LEVEL,
LPAD(last_name,LENGTH(last_name)+(LEVEL*2)-2,'_')
LAST_NAME
FROM employees
CONNECT BY employee_id = PRIOR manager_id;
COLUMN name CLEAR
```

