Oracle Database 10g: SQL Fundamentals II
Student Guide • Volume 2

D17111GC30
Edition 3.0
January 2009
D57874

Development Program (WDP) eKit materials are provided for WDP in-class use only. Copying eKit materials is strictly prohibited and is in violation of Oracle copyright. All WDP students must receive an eKit watermarked with their name and email. Contact OracleWDP_ww@oracle.com if you have not received your personalized eKit.
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
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
### 2 Constraint Management

- Adding a Constraint Syntax  
- Adding a Constraint  
- **ON DELETE CASCADE**  
- Deferring Constraints  
- Dropping a Constraint  
- Disabling Constraints  
- Enabling Constraints  
- Cascading Constraints  
- Overview of Indexes  

<table>
<thead>
<tr>
<th>Constraint Syntax</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON DELETE CASCADE</td>
<td>2-12</td>
</tr>
</tbody>
</table>

### 2.2 CREATE INDEX with the CREATE TABLE Statement

- **CREATE INDEX** with the **CREATE TABLE** Statement  

### 2.3 Function-Based Indexes

- Function-Based Indexes  

### 2.4 Removing an Index

- Removing an Index  
- **DROP TABLE** ... **PURGE**  

### 2.5 FLASHBACK TABLE Statement

- **FLASHBACK TABLE** Statement  

### 2.6 External Tables

- External Tables  
- Creating a Directory for the External Table  
- Creating an External Table  
- Creating an External Table by Using **ORACLE_LOADER**  
- Querying External Tables  
- Summary  

### Practice 2: Overview

- Practice 2: Overview  

### 3 Manipulating Large Data Sets

- Objectives  
- Using Subqueries to Manipulate Data  
- Copying Rows from Another Table  
- Inserting Using a Subquery as a Target  
- Retrieving Data with a Subquery as Source  
- Updating Two Columns with a Subquery  
- Updating Rows Based on Another Table  
- Deleting Rows Based on Another Table  
- Using the **WITH CHECK OPTION** Keyword on DML Statements  
- Overview of the Explicit Default Feature  
- Using Explicit Default Values  
- Overview of Multitable **INSERT** Statements  
- Types of Multitable **INSERT** Statements  
- Multitable **INSERT** Statements  
- Unconditional **INSERT ALL**  
- Conditional **INSERT ALL**  
- Conditional **INSERT FIRST**  

---

Oracle University and SQL Star International Limited use only.

Oracle University and SQL Star International Limited use only.
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
Oracle University and SQL Star International Limited use only

Development Program (WDP) eKit materials are provided for WDP in-class use only. Copying eKit materials is strictly prohibited and is in violation of Oracle copyright. All WDP students must receive an eKit watermarked with their name and email. Contact OracleWDP_ww@oracle.com if you have not received your personalized eKit.
Additional Practices
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
<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>HIRE_DATE</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>201 17-FEB-1996</td>
<td>13000</td>
</tr>
<tr>
<td>2</td>
<td>202 17-AUG-1997</td>
<td>6000</td>
</tr>
<tr>
<td>3</td>
<td>203 07-JUN-1994</td>
<td>6500</td>
</tr>
<tr>
<td>4</td>
<td>204 07-JUN-1994</td>
<td>10000</td>
</tr>
<tr>
<td>5</td>
<td>205 07-JUN-1994</td>
<td>12000</td>
</tr>
<tr>
<td>6</td>
<td>206 07-JUN-1994</td>
<td>8300</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Deptno</th>
<th>Dname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary Key</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>Number</td>
<td>VARCHAR2</td>
</tr>
<tr>
<td>Length</td>
<td>4</td>
<td>30</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>INDEX_NAME</th>
<th>TABLE_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCATIONS_PK_IDX</td>
<td>LOCATIONS_NAMED_INDEX</td>
</tr>
</tbody>
</table>
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 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

<table>
<thead>
<tr>
<th>CITY</th>
<th>DEPARTMENT_NAME</th>
<th>JOB_ID</th>
<th>SUM(SALARY)</th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
<td>null</td>
<td>null</td>
<td>129900</td>
</tr>
<tr>
<td>null</td>
<td>AD_VP</td>
<td>34000</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>AC_MGR</td>
<td>12000</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>FL_MGR</td>
<td>12000</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>AD_PRES</td>
<td>24000</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>AC_ACCOUNT</td>
<td>8300</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>FL_ACCOUNT</td>
<td>39600</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>Finance</td>
<td>51600</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>Finance</td>
<td>12000</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>Finance</td>
<td>39600</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>Executive</td>
<td>58000</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>AD_VP</td>
<td>34000</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>AD_PRES</td>
<td>24000</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>Accounting</td>
<td>20300</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>AC_MGR</td>
<td>12000</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>Accounting</td>
<td>8300</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>AC_ACCOUNT</td>
<td>8300</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>FL_ACCOUNT</td>
<td>39600</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>Finance</td>
<td>51600</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>Finance</td>
<td>12000</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>Finance</td>
<td>39600</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>Executive</td>
<td>58000</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>AD_VP</td>
<td>34000</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>AD_PRES</td>
<td>24000</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>Accounting</td>
<td>20300</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>AC_MGR</td>
<td>12000</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>Accounting</td>
<td>8300</td>
<td></td>
</tr>
<tr>
<td>null</td>
<td>AC_ACCOUNT</td>
<td>8300</td>
<td></td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>DEPARTMENT_ID</th>
<th>JOB_ID</th>
<th>MANAGER_ID</th>
<th>MAX(SALARY)</th>
<th>MIN(SALARY)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(null) AC_MGR</td>
<td>101</td>
<td>12000</td>
<td>12000</td>
</tr>
<tr>
<td>2</td>
<td>(null) SH_CLERK</td>
<td>122</td>
<td>3800</td>
<td>2500</td>
</tr>
<tr>
<td>3</td>
<td>(null) SH_CLERK</td>
<td>124</td>
<td>3100</td>
<td>2600</td>
</tr>
<tr>
<td>4</td>
<td>(null) MK_MAN</td>
<td>100</td>
<td>13000</td>
<td>13000</td>
</tr>
<tr>
<td>5</td>
<td>(null) ST_MAN</td>
<td>100</td>
<td>6200</td>
<td>5000</td>
</tr>
<tr>
<td>6</td>
<td>(null) ST_CLERK</td>
<td>121</td>
<td>3300</td>
<td>2100</td>
</tr>
<tr>
<td>7</td>
<td>(null) SA_REP</td>
<td>148</td>
<td>11500</td>
<td>6100</td>
</tr>
<tr>
<td>8</td>
<td>(null) SH_CLERK</td>
<td>120</td>
<td>3200</td>
<td>2500</td>
</tr>
<tr>
<td>9</td>
<td>(null) AD_ASST</td>
<td>101</td>
<td>4400</td>
<td>4400</td>
</tr>
<tr>
<td>10</td>
<td>(null) AD_PRES</td>
<td>(null)</td>
<td>24000</td>
<td>24000</td>
</tr>
</tbody>
</table>

...
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
     ```sql
     TZ_OFFSET('AUSTRALIA/SYDNEY')
     1 +11:00
     ```
   - Chile/Easter Island
     ```sql
     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.

   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.

   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.

<table>
<thead>
<tr>
<th>#</th>
<th>LAST_NAME</th>
<th>EXTRACT(MONTH FROM HIRE_DATE)</th>
<th>HIRE_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Grant</td>
<td>1</td>
<td>1 13-JAN-2000</td>
</tr>
<tr>
<td>2</td>
<td>De Hsan</td>
<td>1</td>
<td>1 13-JAN-1993</td>
</tr>
<tr>
<td>3</td>
<td>Hunold</td>
<td>1</td>
<td>1 03-JAN-1990</td>
</tr>
<tr>
<td>4</td>
<td>Landry</td>
<td>1</td>
<td>1 14-JAN-1999</td>
</tr>
<tr>
<td>5</td>
<td>Davies</td>
<td>1</td>
<td>1 29-JAN-1997</td>
</tr>
<tr>
<td>6</td>
<td>Partners</td>
<td>1</td>
<td>1 05-JAN-1997</td>
</tr>
<tr>
<td>7</td>
<td>Zlotkey</td>
<td>1</td>
<td>1 29-JAN-2000</td>
</tr>
<tr>
<td>8</td>
<td>Tucker</td>
<td>1</td>
<td>1 30-JAN-1997</td>
</tr>
<tr>
<td>9</td>
<td>King</td>
<td>1</td>
<td>1 30-JAN-1996</td>
</tr>
<tr>
<td>10</td>
<td>Marvins</td>
<td>1</td>
<td>1 24-JAN-2000</td>
</tr>
<tr>
<td>11</td>
<td>Fox</td>
<td>1</td>
<td>1 24-JAN-1998</td>
</tr>
<tr>
<td>12</td>
<td>Johnson</td>
<td>1</td>
<td>1 04-JAN-2000</td>
</tr>
<tr>
<td>13</td>
<td>Taylor</td>
<td>1</td>
<td>1 24-JAN-1998</td>
</tr>
<tr>
<td>14</td>
<td>Sarchand</td>
<td>1</td>
<td>1 27-JAN-1996</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>King</td>
<td>24000</td>
</tr>
<tr>
<td>Kochhar</td>
<td>17000</td>
</tr>
<tr>
<td>De Haan</td>
<td>17000</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>198 O'Connell</td>
</tr>
<tr>
<td>2</td>
<td>199 Grant</td>
</tr>
<tr>
<td>3</td>
<td>120 Weiss</td>
</tr>
<tr>
<td>4</td>
<td>121 Fripp</td>
</tr>
<tr>
<td>5</td>
<td>122 Kaufling</td>
</tr>
<tr>
<td>6</td>
<td>123 Vollman</td>
</tr>
<tr>
<td>7</td>
<td>124 Mourgos</td>
</tr>
<tr>
<td>8</td>
<td>125 Nayer</td>
</tr>
</tbody>
</table>

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.
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`.

<table>
<thead>
<tr>
<th>JOB_TITLE</th>
<th>JOB_TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>President</td>
<td>24000</td>
</tr>
<tr>
<td>Administration Vice President</td>
<td>17000</td>
</tr>
<tr>
<td>Sales Manager</td>
<td>14000</td>
</tr>
<tr>
<td>Marketing Manager</td>
<td>13000</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>HIRE_DATE</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hunold</td>
<td>03-JAN-1990</td>
<td>9000</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>HIRE_DATE</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hunold</td>
<td>03-JAN-1990</td>
<td>9000</td>
</tr>
<tr>
<td>2</td>
<td>Ernst</td>
<td>21-MAY-1991</td>
<td>6000</td>
</tr>
<tr>
<td>3</td>
<td>Austin</td>
<td>25-JUN-1997</td>
<td>4800</td>
</tr>
<tr>
<td>4</td>
<td>Pataballa</td>
<td>05-FEB-1998</td>
<td>4800</td>
</tr>
<tr>
<td>5</td>
<td>Lorentz</td>
<td>07-FEB-1999</td>
<td>4200</td>
</tr>
</tbody>
</table>

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.
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.

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>MANAGER_ID</th>
<th>LEVEL</th>
<th>LAST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>(null)</td>
<td>1 King</td>
</tr>
<tr>
<td>2</td>
<td>101</td>
<td>100</td>
<td>1 Kochhar</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>(null)</td>
<td>2 __King</td>
</tr>
<tr>
<td>4</td>
<td>102</td>
<td>100</td>
<td>1 De Haan</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>(null)</td>
<td>2 __King</td>
</tr>
<tr>
<td>6</td>
<td>103</td>
<td>102</td>
<td>1 Hunold</td>
</tr>
<tr>
<td>7</td>
<td>102</td>
<td>100</td>
<td>2 __De Haan</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>(null)</td>
<td>3 ____King</td>
</tr>
<tr>
<td>310</td>
<td>101</td>
<td>100</td>
<td>2 __Kochhar</td>
</tr>
<tr>
<td>311</td>
<td>100</td>
<td>(null)</td>
<td>3 ____King</td>
</tr>
<tr>
<td>312</td>
<td>206</td>
<td>205</td>
<td>1 Gietz</td>
</tr>
<tr>
<td>313</td>
<td>205</td>
<td>101</td>
<td>2 __Higgins</td>
</tr>
<tr>
<td>314</td>
<td>101</td>
<td>100</td>
<td>3 ____Kochhar</td>
</tr>
<tr>
<td>315</td>
<td>100</td>
<td>(null)</td>
<td>4 ______King</td>
</tr>
</tbody>
</table>

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

   ```sql
   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.

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

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Deptno</th>
<th>Dname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary Key</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>Number</td>
<td>VARCHAR2</td>
</tr>
<tr>
<td>Length</td>
<td>4</td>
<td>30</td>
</tr>
</tbody>
</table>

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
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

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

   - Chile/Easter Island

   ```sql
   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.

   ```sql
   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.

   ```sql
   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.

   ```sql
   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.

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

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

   ```sql
   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.

```sql
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.

```sql
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.
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.

```sql
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.

```sql
ROLLBACK;
```
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;
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
```