**SQL, PL/SQL & TOAD - 2 Weeks (24 hours)**

* **Getting started with SQL**
* **SQL language to create and manipulate SQL database objects.**
* **Database creation and management**
* **Stored Procedures**
* **SQL QUERIES: UNIONS, JOINS, QUERIES, SUBQUERIES**
* **Character String Functions: LPAD, ASCII, RPAD, CHR, LTRIM, CONCAT, RTRIM, INITCAP, TRIM, INSTR,REPLACE, LENGTH, SOUNDEX, LOWER, SUBSTR, UPPER, VSIZE**
* **Aggregate/ Group functions: STDDEV, AVG, SUM, COUNT, VARIANCE, MAX, Group By, MEDIAN, Incorrect Usage, MIN**
* **Date Timestamp Functions: ADD\_MONTHS, ROUND, CURRENT DATE, SESSIONTIMEZONE, CURRENT TIMESTAMP, SYSDATE, DBTIMEZONE, SYSTIMESTAMP, EXTRACT, TO\_CHAR, LAST\_DAY, TO\_DATE, LOCALTIMESTAMP, TO\_TIMESTAMP, MONTHS\_BETWEEN, NEW\_TIME, TRUNC, NEXT\_DAY**
* **Numerical Math Functions: Log, ABS, MOD, POWER, ROUND, CEIL, SQRT, EXP, FLOOR, TRUNC**
* **Conversion Functions: ROWIDTONCHAR, DECODE, TO\_CHAR, NVL, TO\_NUMBER, NVL2, TRANSLATE, ROWIDTOCHAR**
* **Analytical Functions: ROWNUM, CUBE, Group by, HAVING**
* **Miscellaneous Functions: GREATEST, LEAST**
* **Intro to DBMS, RDBMS, Normalization, SQL, PL-SQL**
* **Inserting, Deleting, and Updating Data**
* **Commit and Rollback**
* **DM, DDL, DCL**
* **SELECT, WHERE clauses, Sorting, Wildcards, Distinct**
* **Create, Alter, Drop Tables**
* **Different data types for fields**
* **Constraints: Primary keys, Foreign keys, Unique, Not Null**
* **Group and Have clause**
* **Joins: inner join, outer joins, self join**
* **Introduction to Functions, Procedures, Triggers, Cursor, Sequence**

SELECT table\_name FROM user\_tables;

SELECT COUNT(\*) FROM user\_tables;

SELECT table\_name FROM tabs;

SET linesize 400;

SET pagesize 400;

SELECT \* FROM employees;

SELECT first\_name, last\_name, salary FROM employees;

DESCRIBE employees;

SELECT \* FROM employees

WHERE employee\_id=111;

SELECT \* FROM employees

WHERE first\_name='ismael';

SELECT \* FROM employees

WHERE first\_name='ISMAEL';

SELECT \* FROM employees

WHERE first\_name='Ismael';

SELECT \* FROM employees

WHERE UPPER(first\_name)='ISMAEL';

SELECT \* FROM employees

WHERE LOWER(first\_name)='ismael';

SELECT \* FROM employees

WHERE INITCAP(first\_name)='Ismael';

SELECT \* FROM employees

ORDER BY first\_name;

SELECT \* FROM employees

WHERE UPPER(first\_name)='PETER';

SELECT \* FROM employees

WHERE UPPER(first\_name)='PETER'

AND UPPER(last\_name)='HALL';

SELECT \* FROM employees

WHERE UPPER(first\_name)='PETER'

OR UPPER(last\_name)='HALL';

SELECT \* FROM employees

WHERE UPPER(first\_name) LIKE 'P%';

SELECT salary, commission FROM employees;

Describe employees;

Write a Query to retrieve employees Name, Salary, and Commission.  
SELECT first\_name, last\_name, salary, commission\_pct

FROM employees;

Find out those Employees who have commission.  
SELECT first\_name, last\_name, commission\_pct

FROM employees

WHERE commission\_pct IS NOT NULL

Find out those Employees who do not have commission.  
SELECT first\_name, last\_name, commission\_pct

FROM employees

WHERE commission\_pct IS NULL

SELECT first\_name,last\_name, salary, commission\_pct, salary+salary\*commission\_pct FROM Employees;

SELECT first\_name,last\_name,salary,commission\_pct, salary+salary\*NVL(commission\_pct,0) FROM employees;

SELECT first\_name,last\_name,salary,commission\_pct, salary+salary\*NVL(commission\_pct,.1) FROM employees;

Find out employees monthly & yearly Salary with commission.

SELECT first\_name, last\_name, salary+salary\*NVL(commission\_pct,0) Monthly\_Salary, (salary+salary\*NVL(commission\_pct,0))\*12 Yearly\_Salary

FROM employees;

**Group functions:**

**AVG, COUNT, SUM, MAX, MIN, STDDEV, VARRIANCE**

SELECT MAX(salary)

FROM employees

SELECT first\_name,last\_name

FROM employees

WHERE salary=34560;

SELECT first\_name,last\_name

FROM employees

WHERE salary=(SELECT MAX(salary)

FROM employees);

Find out those employees who get more than average Salary.

SELECT AVG(salary)

FROM employees;

SELECT first\_name,last\_name,salary

FROM employees

WHERE salary>6568.13084;

SELECT first\_name,last\_name,salary

FROM employees

WHERE salary>(SELECT AVG(salary)

FROM employees);

Find out who earns the maximum salary

SELECT first\_name, last\_name, salary

FROM employees

WHERE salary= (SELECT MAX(salary) FROM employees);

Find out those employees who get the same salary as Peter Tucker  
SELECT first\_name,last\_name,salary

FROM employees

WHERE UPPER(first\_name)=’PETER’

AND UPPER(last\_name)=’TUCKER’;

SELECT first\_name,last\_name,salary

FROM employees

WHERE salary=10000;

SELECT first\_name,last\_name,salary

FROM employees

WHERE salary=(SELECT salary

FROM employees

WHERE UPPER(first\_name)=’PETER’

AND UPPER(last\_name)=’TUCKER’);

Write a Query to retrieve to display first 10 records from a table.

SELECT \*

FROM employees

WHERE rownum<11;

SELECT \*

FROM employees

WHERE rownum<=10;

Find out row number 11 to 20 from of a table.

(SELECT \*

FROM employees

WHERE rownum<21)

MINUS

(SELECT \*

FROM employees

WHERE rownum<11);

(SELECT \*

FROM employees

WHERE rownum<=20)

MINUS

(SELECT \*

FROM employees

WHERE rownum<=10);

Write a query to display the last 10 records.

SELECT \*

FROM (SELECT \* FROM employees order by rownum DESC)

WHERE rownum<=10

ORDER BY employee\_id;

Find out Top 10 salaried employees.  
 Solution 1:

Step1:

CREATE TABLE employees\_top\_ten\_salaried

AS

SELECT first\_name, last\_name, salary

FROM employees

ORDER BY salary DESC;

Step2:

SELECT first\_name, last\_name, salary

FROM employees\_top\_ten\_salaried\_new

WHERE rownum<=10;

Solution2:

SELECT first\_name, last\_name, salary

FROM (SELECT first\_name, last\_name, salary FROM employees ORDER BY salary DESC)

WHERE rownum<=10;

Solution3:

SELECT salary FROM employees X

WHERE 10>(SELECT COUNT(\*) FROM employees Y WHERE X.salary<Y.salary)

ORDER BY salary DESC

SOLUTION 4:

SELECT first\_name, salary, salary\_rank

FROM (SELECT first\_name, salary,RANK() OVER (ORDER BY salary DESC) salary\_rank FROM employees)

WHERE salary\_rank<=10

ORDER BY salary DESC;

Find out LEAST 10 salaried employees.

SELECT salary FROM employees X

WHERE 10>(SELECT COUNT(\*) FROM employees Y WHERE X.salary>Y.salary)

ORDER BY salary DESC

**JOINING**

Find out Employees Name, Salary, department\_id.

SELECT first\_name, last\_name, salary, department\_id

FROM employees;  
  
  
Find out Employees Name, Salary, and department name.

SELECT employees.first\_name, employees.last\_name, employees.salary, departments.department\_name

FROM employees, departments

WHERE employees.department\_id= departments.department\_id;

SELECT e.first\_name, e.last\_name, e.salary, d.department\_name

FROM employees e, departments d

WHERE e.department\_id= d.department\_id;

Find out Employees Name, Salary, department name, City.

SELECT e.first\_name, e.last\_name, e.salary, d.department\_name, l.city

FROM employees e, departments d, locations l

WHERE e.department\_id= d.department\_id

AND d.location\_id=l.location\_id;

Find out Employees Name, Salary, department name, City, Country Name.   
  
SELECT e.first\_name, e.last\_name, e.salary, d.department\_name, l.city, c.country\_name

FROM employees e, departments d, locations l, countries c

WHERE e.department\_id= d.department\_id

AND d.location\_id=l.location\_id

AND l.country\_id= c.country\_id;

Find out Employees Name, Salary, department name, City, Country Name, Region name.   
  
  
Find out Employees Name, Salary, department name, City, Country Name, Region name, start date.   
  
  
Find out Employees Name, Salary, department name, City, Country Name, Region name, start date,Job Title.

SELECT e.first\_name, e.last\_name, e.salary,d.department\_name,l.city,c.country\_name,r.region\_name,jh.start\_date,j.job\_title

FROM employees e,departments d,locations l, countries c, regions r,job\_history jh, jobs j

WHERE e.department\_id =d.department\_id

AND d.location\_id =l.location\_id

AND l.country\_id =c.country\_id

AND c.region\_id =r.region\_id

AND jh.department\_id=d.department\_id

AND jh.job\_id =j.job\_id

Find out the departments which have employees.

OR

Find out those employees who belong to any Department.

SELECT e.first\_name,e.last\_name, e.department\_id, d.department\_name

FROM employees e, departments d

WHERE e.department\_id=d.department\_id;

Find out the departments which have employees. (Matching records)

SELECT e.first\_name,e.last\_name,e.department\_id, d.department\_name

FROM employees e, departments d

WHERE e.department\_id=d.department\_id

Find out the departments which have employees including the departments that do not have employees. (Matching & non-matching records)

SELECT e.first\_name,e.last\_name,e.department\_id, d.department\_name

FROM employees e, departments d

WHERE e.department\_id(+)=d.department\_id

Find out the departments which do not have employees.

(SELECT e.first\_name,e.last\_name,e.department\_id, d.department\_name

FROM employees e, departments d

WHERE e.department\_id(**+**)=d.department\_id)

MINUS

(SELECT e.first\_name,e.last\_name,e.department\_id, d.department\_name

FROM employees e, departments d

WHERE e.department\_id=d.department\_id);

Find out the employees who belong to a department. (Matching records)

SELECT e.first\_name,e.last\_name,e.department\_id, d.department\_name

FROM employees e, departments d

WHERE e.department\_id=d.department\_id

Find out those employees who belong to a department including the employees who do not belong to a department. (Matching and non-matching records)

SELECT e.first\_name,e.last\_name,e.department\_id, d.department\_name

FROM employees e, departments d

WHERE e.department\_id=d.department\_id(**+**)

Find out those employees who do not belong to any Department.

(SELECT e.first\_name,e.last\_name,e.department\_id, d.department\_name

FROM employees e, departments d

WHERE e.department\_id=d.department\_id(**+**))

MINUS

(SELECT e.first\_name,e.last\_name,e.department\_id, d.department\_name

FROM employees e, departments d

WHERE e.department\_id=d.department\_id);

Find out the departments which do not have employees AND find out those employees who do not belong to any Department.

((SELECT e.first\_name,e.last\_name,e.department\_id, d.department\_name

FROM employees e, departments d

WHERE e.department\_id(+)=d.department\_id)

MINUS

(SELECT e.first\_name,e.last\_name,e.department\_id, d.department\_name

FROM employees e, departments d

WHERE e.department\_id=d.department\_id))

UNION

((SELECT e.first\_name,e.last\_name,e.department\_id, d.department\_name

FROM employees e, departments d

WHERE e.department\_id=d.department\_id(+))

MINUS

(SELECT e.first\_name,e.last\_name,e.department\_id, d.department\_name

FROM employees e, departments d

WHERE e.department\_id=d.department\_id))

**SELF JOIN**

Find out managers of the employees.

SELECT e.first\_name employee\_first\_name, e.last\_name employee\_last\_name, m.first\_name manager\_first\_name, m.last\_name manager\_last\_name

FROM employees e, employees m

WHERE m.employee\_id=e.manager\_id;

SELECT e.first\_name employee\_first\_name, e.last\_name employee\_last\_name, m.first\_name manager\_first\_name, m.last\_name manager\_last\_name

FROM employees e, employees m

WHERE m.employee\_id=e.manager\_id

ORDER BY m.first\_name;

Find out the manager of the Shanta Vollman.

SELECT e.first\_name employee\_first\_name, e.last\_name employee\_last\_name, m.first\_name manager\_first\_name, m.last\_name manager\_last\_name

FROM employees e, employees m

WHERE m.employee\_id=e.manager\_id

AND UPPER(e.first\_name)='SHANTA'

AND UPPER(e.last\_name)='VOLLMAN';

**GROUP FUNCTIONS: SUM, MAX, MIN, AVG, STDDEV, VARIANCE, COUNT**

**WHERE**

**GROUP BY**

**HAVING**

Find out those managers who manage only one employee.

SELECT e.manager\_id, COUNT(e.employee\_id)

FROM employees e, employees m

WHERE m.employee\_id=e.manager\_id

GROUP BY e.manager\_id

HAVING COUNT(e.manager\_id)=1;

Find out those managers who manage the most employees

Find out those managers who manage the least employees

Find out employees average Salary  
SELECT AVG(salary)

FROM employees;  
  
Find out employees average Salary within their Department  
SELECT AVG(salary)

FROM employees

GROUP BY department\_id;

SELECT department\_id, AVG(salary)

FROM employees

WHERE department\_id IS NOT NULL

GROUP BY department\_id;

SELECT department\_id, AVG(salary)

FROM employees

WHERE department\_id IS NOT NULL

GROUP BY department\_id

ORDER BY department\_id;

SELECT department\_id, AVG(salary)

FROM employees

WHERE department\_id IS NOT NULL

GROUP BY department\_id

ORDER BY department\_id ASC;

SELECT department\_id,AVG(salary)

FROM employees

WHERE department\_id IS NOT NULL

GROUP BY department\_id

ORDER BY department\_id DESC;

Find out employees average Salary greater than 10,000 within their Department  
  
SELECT department\_id, AVG(salary)

FROM employees

WHERE department\_id IS NOT NULL

GROUP BY department\_id

HAVING AVG(salary)>10000

ORDER BY department\_id DESC;

14. Write a SQL Query,  
  
  
                If’ job is IT\_PROG, the Salary increases 10%  
  
  
                If job is FI\_ACCOUNT, the Salary increases 15%  
  
  
                If job is ST\_CLERK, the Salary increases 20%  
  
  
                If job is SA\_REP, the Salary increases 25%  
  
  
                If job is AC\_ACCOUNT, the Salary increases 30%  
  
  
                For all other roles, there is no increase Salary

SELECT first\_name, last\_name, job\_id, salary,

DECODE (job\_id, 'IT\_PROG', salary\*1.10,

'FI\_ACCOUNT', salary\*1.15,

'ST\_CLERK', salary\*1.20,

'SA\_REP', salary\*1.25,

'AC\_ACCOUNT', salary\*1.30,

salary) increased\_salary

FROM employees;

Display the salary of the employees whose job title are as follows: 'IT\_PROG', 'FI\_ACCOUNT', 'ST\_CLERK','SA\_REP', 'AC\_ACCOUNT'.

SELECT first\_name, last\_name, job\_id, salary, increased\_salary

FROM

WHERE job\_id IN ('IT\_PROG','FI\_ACCOUNT','ST\_CLERK','SA\_REP','AC\_ACCOUNT');

Display those employees whose salary got increased.   
  
SELECT first\_name, last\_name, job\_id, salary, increased\_salary

FROM (SELECT first\_name, last\_name, job\_id, salary,

DECODE (job\_id, 'IT\_PROG', salary\*1.10,

'FI\_ACCOUNT', salary\*1.15,

'ST\_CLERK', salary\*1.20,

'SA\_REP', salary\*1.25,

'AC\_ACCOUNT', salary\*1.30,

salary) increased\_salary

FROM employees)

WHERE job\_id IN ('IT\_PROG','FI\_ACCOUNT','ST\_CLERK','SA\_REP','AC\_ACCOUNT');

Other way:

SELECT first\_name, last\_name, job\_id,salary,

DECODE (job\_id, 'IT\_PROG', salary\*1.10,

'FI\_ACCOUNT', salary\*1.15,

'ST\_CLERK', salary\*1.20,

'SA\_REP', salary\*1.25,

'AC\_ACCOUNT', salary\*1.30) increased\_salary

FROM employees

WHERE job\_id IN ('IT\_PROG','FI\_ACCOUNT','ST\_CLERK','SA\_REP','AC\_ACCOUNT');

Display the number of Distinct Department in the Employee table

SELECT COUNT(DISTINCT(department\_id))

FROM employees;

Display all the Distinct Departments in the Employee table

SELECT DISTINCT(department\_id)

FROM employees

WHERE department\_id IS NOT NULL

Create a Table with employees\_your first name using Employee Table with all records

CREATE TABLE employees\_abu

AS

SELECT \* FROM employees;

Create a Table with employees\_your last name using Employee Table with no records- CREATING AN EMPTY TABLE

CREATE TABLE employees\_hanip

AS

SELECT \* FROM employees

WHERE 1=2;

Insert a row in that Table you created  
  
INSERT INTO employees\_hanip (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

VALUES (1000, 'Abu', 'Hanip', 'abuhanip@pnt.edu', '718-878-4709', '31-Jan-09', 'QA-TESTER', 90000, .5, 100, 500);

Create a text file, put 5 records and execute it in the Command Line

* Open a Notepad
* Write the following lines in the Notepad:

INSERT INTO employees\_hanip (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

VALUES (556, 'Russo1', 'MMK', 'russo1@pnt.edu', '718-878-4709', '31-Jan-09', 'QA-TESTER', 90000, .5, 100, 501);

INSERT INTO employees\_hanip (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

VALUES (557, 'Russo2', 'MMK', 'russo2@pnt.edu', '718-878-4709', '31-Jan-09', 'QA-TESTER', 90000, .5, 100, 502);

INSERT INTO employees\_hanip (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

VALUES (558, 'Russo3', 'MMK', 'russo3@pnt.edu', '718-878-4709', '31-Jan-09', 'QA-TESTER', 90000, .5, 100, 503);

INSERT INTO employees\_hanip (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

VALUES (559, 'Russo4', 'MMK', 'russo4@pnt.edu', '718-878-4709', '31-Jan-09', 'QA-TESTER', 90000, .5, 100, 504);

INSERT INTO employees\_hanip (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

VALUES (560, 'Russo5', 'MMK', 'russo5@pnt.edu', '718-878-4709', '31-Jan-09', 'QA-TESTER', 90000, .5, 100, 505);

* Save as sql\_insert\_file.txt in the c:/ drive.
* Run that file in SQL command line as @c:/SQL\_insert\_file.txt
* Use SELECT statement to confirm the entry in the table:

SELECT \* FROM employees\_hanip;

Update a row with that Table you created  
  
UPDATE employee\_hanip

SET phone\_number='718-825-7140'

WHERE employee\_id =1000;

Delete a row from that Table you created

DELETE FROM employees\_hanip

WHERE employee\_id=560;  
  
  
Create a view for employees Name and Phone Number from employees Table.

CREATE VIEW employees\_view

AS

SELECT first\_name, last\_name, phone\_number

FROM employees;

Create and account with [your name] and assign all those privileges.

DML-SELECT, INSERT, UPDATE, DELETE

DDL-DR CAT- DROP, RENAME, CREATE, ALTER,TRUNCATE

DCL-REVOKE, GRANT

DELETE, DROP, TRUNCATE

DELETE FROM employees\_hanip1

WHERE employee\_id=1000;

DROP TABLE employees\_hanip2;

TRUNCATE TABLE employees\_hanip3;

SELECT last\_name,MONTHS\_BETWEEN(SYSDATE,hire\_date) Total\_months, MONTHS\_BETWEEN(SYSDATE,hire\_date)/12 Total\_years

FROM Employees;

SELECT last\_name,TRUNC(MONTHS\_BETWEEN(SYSDATE,hire\_date)) Total\_months, TRUNC(MONTHS\_BETWEEN(SYSDATE,hire\_date)/12) Total\_years

FROM employees;

SELECT last\_name,TRUNC(MONTHS\_BETWEEN(SYSDATE,hire\_date)/12) years,TRUNC(MOD(MONTHS\_BETWEEN(SYSDATE,hire\_date),12)) months,

FROM employees;

SELECT RPAD(last\_name,11)|| ' has been working '||LPAD(TRUNC(MONTHS\_BETWEEN(SYSDATE,hire\_date)/12),2,'0') ||' years and '|| LPAD(TRUNC(MOD(MONTHS\_BETWEEN(SYSDATE,hire\_date),12)),2,'0')|| ' months.'

FROM employees;

SELECT last\_name

FROM employees

WHERE UPPER(last\_name) LIKE 'J%';

SELECT last\_name

FROM employees

WHERE UPPER(last\_name) LIKE 'K%';

SELECT last\_name

FROM employees

WHERE UPPER(last\_name) LIKE 'L%';

SELECT last\_name

FROM employees

WHERE UPPER(last\_name) LIKE 'M%';

SELECT last\_name FROM employees

WHERE last\_name LIKE 'J%' OR last\_name LIKE 'K%' OR last\_name LIKE 'L%' OR last\_name LIKE 'M%'

ORDER BY last\_name;

SELECT last\_name, salary

FROM employees

WHERE last\_name LIKE IN ('J%','K%', 'L%','M%')

ORDER BY last\_name

SELECT last\_name,salary

FROM employees

WHERE SUBSTR(last\_name,1,1) IN('J','K','L','M')

Reference: SUBSTR

What is SQL SUBSTRING?

The SQL Substring feature is a function that enables parts of strings to be accessed.

For example, the function SUBSTRING('212-555-1234', 9 , 4) returns:

'1234'

It returns 4 characters starting in position 9.

The Substring function in SQL is used to grab a portion of the stored data. This function is called differently for the different databases:

* MySQL: SUBSTR(), SUBSTRING()
* Oracle: SUBSTR()
* SQL Server: SUBSTRING(

SELECT SUBSTR(last\_name,1,2), salary FROM employees;

SELECT last\_name,SUBSTR(last\_name,1,2), salary FROM employees;

SELECT last\_name,SUBSTR(last\_name,1,3), salary FROM employees;

SELECT last\_name,SUBSTR(last\_name,1,1), salary FROM employees;

SELECT last\_name,SUBSTR(last\_name,1,1), SUBSTR(last\_name,2,1),salary FROM employees;

SELECT last\_name, SUBSTR(last\_name, -1), SUBSTR(last\_name, 1)

FROM employees

(SELECT last\_name FROM employees WHERE last\_name LIKE 'J%')

UNION

(SELECT last\_name FROM employees WHERE last\_name LIKE 'K%')

UNION

(SELECT last\_name FROM employees WHERE last\_name LIKE 'L%')

UNION

(SELECT last\_name FROM employees WHERE last\_name LIKE 'M%');

(SELECT last\_name FROM employees WHERE last\_name LIKE 'J%')

UNION ALL

(SELECT last\_name FROM employees WHERE last\_name LIKE 'K%')

UNION ALL

(SELECT last\_name FROM employees WHERE last\_name LIKE 'L%')

UNION ALL

(SELECT last\_name FROM employees WHERE last\_name LIKE 'M%');

**The SQL UNION Operator**

The UNION operator is used to combine the result-set of two or more SELECT statements.

The UNION query allows you to combine the result sets of 2 or more "select" queries. It removes duplicate rows between the various "select" statements.

Each SQL statement within the UNION query must have the same number of fields in the result sets with similar data types.

SQL Union is used to combine results of two SQL queries into one. The syntax of SQL UNION is as follows:

**SELECT** column\_list1 **FROM** table1

**UNION** (**ALL**)

**SELECT** column\_list2 **FROM** table2

By default SQL UNION eliminates all duplicate records, in this case NULL values are considered as a single value. To enable duplication of records, you can use ALL keyword followed after UNION explicitly.

**A= {1, 3, 4, 6}**

**B= {2, 3, 5}**

**AB={common & uncommon}**

**AB={3,1,4,6,2,5}**

**={1,2,3,4,5,6}**

**A UNION B={1,2,3,4,5,6}**

**A UNION ALL B={ common, uncommon & duplicate}**

**A UNION ALL B={3,1,4,6,2,5,3}**

**A UNION ALL B={1,2,3,3,4,5,6}**

**A= {1, 3, 4, 6}**

**B= {2, 3, 5}**

**AB={common}**

**AB={3}**

**A INTERSECT B ={3}**

**A= {1, 3, 4, 6}**

**B= {2, 3, 5}**

**A MINUS B={1,4,6 }**

**B MINUS A={2, 5}**

**CREATE OR REPLACE PROCEDURE raise\_salary**

**(id NUMBER)**

**IS**

**BEGIN**

**UPDATE employees**

**SET salary=salary+(salary\*10/100)**

**WHERE employee\_id =id;**

**END raise\_salary;**

**CREATE OR REPLACE PROCEDURE raise\_salary**

**(id NUMBER,**

**increased\_percent NUMBER)**

**IS**

**BEGIN**

**UPDATE employees**

**SET salary=salary+(salary\*increased\_percent/100)**

**WHERE employee\_id =id;**

**END raise\_salary;**

PL/SQL

Write a Function to calculate employees Income Tax where Income Tax is .05%  
CREATE OR REPLACE FUNCTION INCOME\_TAX (salary IN NUMBER)

RETURN NUMBER IS

BEGIN

RETURN (salary \* .05);

END INCOME\_TAX;

What is Trigger?

Overview of Triggers

A trigger is a named program unit that is stored in the database and **fired** (executed) in response to a specified event. The specified **event** is associated with either a table, a view, a schema, or the database, and it is one of the following:

* A database manipulation (DML) statement (DELETE, INSERT, or UPDATE)
* A database definition (DDL) statement (CREATE, ALTER, or DROP)
* A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN)

The trigger is said to be **defined on** the table, view, schema, or database.

Topics:

* [Trigger Types](http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28370/triggers.htm)
* [Trigger States](http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28370/triggers.htm)
* [Data Access for Triggers](http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28370/triggers.htm)
* [Uses of Triggers](http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28370/triggers.htm)

Trigger Types

A **DML trigger** is fired by a DML statement, a **DDL trigger** is fired by a DDL statement, a **DELETE** **trigger** is fired by a DELETE statement, and so on.

An **INSTEAD** **OF** **trigger** is a DML trigger that is defined on a view (not a table). The database fires the INSTEAD OF trigger instead of executing the triggering DML statement.

A **system trigger** is defined on a schema or the database. A trigger defined on a schema fires for each event associated with the owner of the schema (the current user). A trigger defined on a database fires for each event associated with all users.

A **simple trigger** can fire at exactly one of the following **timing points**:

* Before the triggering statement executes
* After the triggering statement executes
* Before each row that the triggering statement affects
* After each row that the triggering statement affects

A **compound trigger** can fire at more than one timing point. Compound triggers make it easier to program an approach where you want the actions you implement for the various timing points to share common data.

Trigger States

A trigger can be in either of two states:

**Enabled**. An enabled trigger executes its trigger body if a triggering statement is entered and the trigger restriction (if any) evaluates to TRUE.

**Disabled**. A disabled trigger does not execute its trigger body, even if a triggering statement is entered and the trigger restriction (if any) evaluates to TRUE.

By default, a trigger is created in enabled state. To create a trigger in disabled state, use the DISABLE clause of the CREATE TRIGGER statement.

Write an Insert Trigger for employees table

CREATE TRIGGER insert\_trigger

AFTER INSERT

ON employeestable1

FOR EACH ROW

BEGIN

INSERT INTO employeestable2 ( EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME,

EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT,

MANAGER\_ID, DEPARTMENT\_ID)

VALUES ( :new.EMPLOYEE\_ID, :new.FIRST\_NAME, :new.LAST\_NAME,

:new.EMAIL, :new.PHONE\_NUMBER, :new.HIRE\_DATE, :new.JOB\_ID,

:new.SALARY, :new.COMMISSION\_PCT, :new.MANAGER\_ID,

:new.DEPARTMENT\_ID);

END;  
  
Write an Update Trigger for employees table  
CREATE TRIGGER update\_trigger

BEFORE UPDATE

ON employeestable2

FOR EACH ROW

BEGIN

UPDATE employeestable3

SET EMPLOYEE\_ID=:new.EMPLOYEE\_ID,

FIRST\_NAME= :new.FIRST\_NAME,

LAST\_NAME=:new.LAST\_NAME,

EMAIL=:new.EMAIL,

PHONE\_NUMBER=:new.PHONE\_NUMBER,

HIRE\_DATE=:new.HIRE\_DATE,

JOB\_ID=:new.JOB\_ID,

SALARY=:new.SALARY,

COMMISSION\_PCT=:new.COMMISSION\_PCT,

MANAGER\_ID=:new.MANAGER\_ID,

DEPARTMENT\_ID=:new.DEPARTMENT\_ID

WHERE employee\_id=:old.employee\_id;

END;  
  
Write a Delete Trigger for employees table  
  
  
Write a procedure which raises employee's salary up to 10%  
  
**CREATE OR REPLACE PROCEDURE raise\_salary**

**(id NUMBER)**

**IS**

**BEGIN**

**UPDATE employees**

**SET salary=salary+(salary\*10/100)**

**WHERE employee\_id =id;**

**END raise\_salary;**

Write a procedure which raises employee's salary as given percentage.

**CREATE OR REPLACE PROCEDURE raise\_salary**

**(id NUMBER,**

**increased\_percent NUMBER)**

**IS**

**BEGIN**

**UPDATE employees**

**SET salary=salary+(salary\*increased\_percent/100)**

**WHERE employee\_id =id;**

**END raise\_salary;**

SELECT e.first\_name,e.last\_name,e.salary,

d.department\_name

FROM employees e, departments d

WHERE e.department\_id=d.department\_id;

SELECT e.first\_name,e.last\_name,e.salary,

d.department\_name,l.city

FROM employees e, departments d,locations l

WHERE e.department\_id=d.department\_id

AND d.location\_id=l.location\_id;

SELECT e.first\_name,e.last\_name,e.salary,

d.department\_name,l.city,c.country\_name

FROM employees e, departments d,locations l,countries c

WHERE e.department\_id=d.department\_id

AND d.location\_id=l.location\_id

AND c.country\_id=l.country\_id;

SELECT e.first\_name,e.last\_name,e.salary,

d.department\_name,l.city,c.country\_name,r.region\_name

FROM employees e, departments d,locations l,countries

c,regions r

WHERE e.department\_id=d.department\_id

AND d.location\_id=l.location\_id

AND c.country\_id=l.country\_id

AND r.region\_id=c.region\_id;

SELECT e.first\_name,e.last\_name,e.salary,

d.department\_name,l.city,c.country\_name,r.region\_name,jh

.start\_date

FROM employees e, departments d,locations l,countries

c,regions r,job\_history jh

WHERE e.department\_id=d.department\_id

AND d.location\_id=l.location\_id

AND c.country\_id=l.country\_id

AND r.region\_id=c.region\_id

AND e.employee\_id=jh.employee\_id;

SELECT e.first\_name,e.last\_name,e.salary,

d.department\_name,l.city,c.country\_name,r.region\_name,jh

.start\_date,j.job\_title

FROM employees e, departments d,locations l,countries

c,regions r,job\_history jh,jobs j

WHERE e.department\_id=d.department\_id

AND d.location\_id=l.location\_id

AND c.country\_id=l.country\_id

AND r.region\_id=c.region\_id

AND e.employee\_id=jh.employee\_id

AND j.job\_id=jh.job\_id;

**Page Format:**

SQL> select \* from employees;

SQL> set pagesize 300

SQL> set linesize 300

**CONCAT:**

SQL> SELECT CONCAT('ABU','HANIP')

2 FROM dual;

SQL> SELECT 'ABU'||'HANIP'

2 FROM dual;

SQL> SELECT CONCAT('ABU','HANIP','CHY')

2 FROM dual;

SQL> SELECT 'ABU'||'HANIP'||'CHY'

2 FROM dual;

SQL> SELECT 'ABU'||' '||'HANIP'||' '||'CHY'

2 FROM dual;

SQL> SELECT 'ABU'||','||'HANIP'||','||'CHY'

2 FROM dual;

SQL> SELECT first\_name,last\_name,salary FROM employees;

SQL> SELECT first\_name||last\_name||salary FROM employees

SQL> SELECT first\_name||' '||last\_name||' is earning '||salary||' per year.' FROM employees

**RPAD:**

SQL> SELECT RPAD(first\_name,15)||' '||RPAD(last\_name,15)||' is earning '||salary||' per year.' FROM employees

**LPAD:**

SQL> SELECT RPAD(first\_name,12)||' '||RPAD(last\_name,12)||' is earning '||LPAD(salary,6)||' per year.' FROM employees

To\_CHAR:

SQL> SELECT RPAD(first\_name,12)||' '||RPAD(last\_name,12)||' is earning '||LPAD(TO\_CHAR(salary,'$9,999.00'),6)||' per year.' FROM employees

SQL> SELECT RPAD(first\_name,12)||' '||RPAD(last\_name,12)||' is earning '||LPAD(TO\_CHAR(salary,'$9,99,999.00'),6)||' per year.' FROM employees

SQL> SELECT RPAD(first\_name,12)||' '||RPAD(last\_name,12)||' is earning '||LPAD(TO\_CHAR(salary,'$9,999.00'),10)||' per year.' FROM employees

SQL> SELECT RPAD(first\_name,12)||' '||RPAD(last\_name,12)||' is earning '||LPAD(TO\_CHAR(salary,'$9,99,999.00'),10)||' per year.' FROM employees

SQL> SELECT RPAD(first\_name,12)||' '||RPAD(last\_name,12)||' is earning '||LPAD(TO\_CHAR(salary,'$9,99,999'),10)||' per year.' FROM employees

**LENGTH:**

SELECT LENGTH('ABUHANIP') FROM dual

SELECT first\_name,salary

FROM employees

WHERE LENGTH(first\_name)=6

**UPPER:**

SELECT first\_name,last\_name,salary

FROM employees

WHERE UPPER(job\_id) LIKE 'SA\_%'

1 select \* FROM employees

2\* WHERE job\_id='Ad\_pres'

1 select \* FROM employees

2\* WHERE UPPER(job\_id)='AD\_PRES'

1 select \* FROM employees

2\* WHERE LOWER(job\_id)='ad\_pres'

3 /

1 select \* FROM employees

2\* WHERE INITCAP(job\_id)='Ad\_pres'

3 /

1 select \* FROM employees

2\* WHERE job\_id='AD\_PRES'

SELECT first\_name,last\_name,job\_id,salary

FROM employees

WHERE UPPER(job\_id) LIKE 'SA\_%'

**SUBSTR:**

SELECT first\_name,last\_name,job\_id,salary

FROM employees

WHERE UPPER(SUBSTR(job\_id,1,3)) = 'SA\_%'

SELECT first\_name,last\_name,job\_id,salary

FROM employees

WHERE UPPER(SUBSTR(job\_id,1,3)) = 'SA\_'

**SYSDATE**

SELECT SYSDATE today, SYSDATE-1 yesterday, SYSDATE+1 tomorrow FROM dual

SELECT SYSDATE, SYSDATE-1, SYSDATE+1 FROM dual

**NVL:**

SQL> SELECT first\_name,last\_name,salary,salary+salary\*commission\_pct

FROM employees;

SELECT first\_name,last\_name,salary,commission\_pct,salary+salary\*commission\_pct

FROM employees

SELECT first\_name,last\_name,salary,

commission\_pct,

salary+salary\*NVL(commission\_pct,0)

FROM employees

SELECT first\_name,last\_name,salary,

commission\_pct,

salary+salary\*NVL(commission\_pct,0) yearly\_salary

FROM employees

AVG, MAX, MIN, SUM,

SQL> SELECT AVG(salary),MAX(salary),MIN(salary),SUM(salary)

FROM employees;

SQL> SELECT COUNT(\*) FROM employees;

SELECT first\_name,last\_name,salary FROM employees

WHERE salary>AVG(salary)

SQL> SELECT AVG(salary) FROM employees;

AVG(SALARY)

-----------

6461.68224

SQL> SELECT first\_name,last\_name,salary

FROM employees

WHERE salary>6461.68224;

SELECT first\_name,last\_name,salary FROM employees

WHERE salary>6461.68224

SELECT first\_name,last\_name,salary

FROM employees

WHERE salary>(SELECT AVG(salary) FROM employees)

SELECT first\_name,last\_name,salary

FROM employees

WHERE salary>(SELECT salary FROM employees

WHERE UPPER(first\_name)='STEVEN'

AND UPPER(last\_name)='KING')

SQL> /

no rows selected

SELECT first\_name,last\_name,salary

FROM employees

WHERE salary>(SELECT salary FROM employees

WHERE UPPER(first\_name)='ELLEN'

AND UPPER(last\_name)='ABEL')

1 SELECT first\_name,last\_name,salary

2 FROM employees

3 WHERE salary=(SELECT salary FROM employees

4 WHERE UPPER(first\_name)='NANCY'

5\* AND UPPER(last\_name)='GREENBERG'

1 SELECT COUNT(department\_id)

2\* FROM EMPLOYEES

SQL> /

COUNT(DEPARTMENT\_ID)

--------------------

106

SQL> ed

Wrote file afiedt.buf

1 SELECT COUNT(DISTINCT(department\_id))

2\* FROM EMPLOYEES

SQL> SELECT SYSDATE FROM dual;

SQL> SELECT first\_name,last\_name,salary,department\_id

2 FROM employees;

SQL> describe departments;

SQL> SELECT \* from DEPARTMENTS;

1 SELECT first\_name,last\_name,department\_name

2\* FROM employees, departments

1 SELECT employees.first\_name,employees.last\_name,departments.department\_name

2 FROM employees, departments

3\* WHERE employees.department\_id=departments.department\_id

1 SELECT e.first\_name,e.last\_name,d.department\_name

2 FROM employees e, departments d

3\* WHERE e.department\_id=d.department\_id

1 SELECT e.first\_name,e.last\_name,d.department\_name

2 FROM employees e, departments d

3\* WHERE e.department\_id=d.department\_id(+)

1 (SELECT e.first\_name,e.last\_name,d.department\_name

2 FROM employees e, departments d

3 WHERE e.department\_id=d.department\_id(+))

4 -

5 (SELECT e.first\_name,e.last\_name,d.department\_name

6 FROM employees e, departments d

7\* WHERE e.department\_id=d.department\_id)

8 /

1 (SELECT e.first\_name,e.last\_name,d.department\_name

2 FROM employees e, departments d

3 WHERE e.department\_id=d.department\_id(+))

4 MINUS

5 (SELECT e.first\_name,e.last\_name,d.department\_name

6 FROM employees e, departments d

7\* WHERE e.department\_id=d.department\_id)

8 /

1 SELECT e.first\_name,e.last\_name,d.department\_name

2 FROM employees e, departments d

3\* WHERE e.department\_id(+)=d.department\_id

4 /

1 (SELECT e.first\_name,e.last\_name,d.department\_name

2 FROM employees e, departments d

3 WHERE e.department\_id(+)=d.department\_id)

4 MINUS

5 (SELECT e.first\_name,e.last\_name,d.department\_name

6 FROM employees e, departments d

7\* WHERE e.department\_id=d.department\_id)

8 /

1 ((SELECT e.first\_name,e.last\_name,d.department\_name

2 FROM employees e, departments d

3 WHERE e.department\_id(+)=d.department\_id)

4 MINUS

5 (SELECT e.first\_name,e.last\_name,d.department\_name

6 FROM employees e, departments d

7 WHERE e.department\_id=d.department\_id))

8 UNION

9 ((SELECT e.first\_name,e.last\_name,d.department\_name

10 FROM employees e, departments d

11 WHERE e.department\_id=d.department\_id(+))

12 MINUS

13 (SELECT e.first\_name,e.last\_name,d.department\_name

14 FROM employees e, departments d

15\* WHERE e.department\_id=d.department\_id))

16 /

1 (SELECT e.first\_name,e.last\_name,d.department\_name

2 FROM employees e, departments d

3 WHERE e.department\_id(+)=d.department\_id)

4 UNION

5 (SELECT e.first\_name,e.last\_name,d.department\_name

6 FROM employees e, departments d

7\* WHERE e.department\_id=d.department\_id(+))

1 (SELECT e.first\_name,e.last\_name,d.department\_name

2 FROM employees e, departments d

3 WHERE e.department\_id(+)=d.department\_id)

4 UNION ALL

5 (SELECT e.first\_name,e.last\_name,d.department\_name

6 FROM employees e, departments d

7\* WHERE e.department\_id=d.department\_id(+))

SQL> SELECT first\_name,last\_name,salary

2 FROM employees;

1 SELECT e.first\_name,e.last\_name,e.salary, d.department\_name

2 FROM employees e, departments d

3\* WHERE e.department\_id=d.department\_id

1 SELECT e.first\_name,e.last\_name,e.salary, d.department\_name,l.city

2 FROM employees e, departments d,locations l

3 WHERE e.department\_id=d.department\_id

4\* AND d.location\_id=l.location\_id

1 SELECT e.first\_name,e.last\_name,e.salary, d.department\_name,l.city,c.country\_name

2 FROM employees e, departments d,locations l,countries c

3 WHERE e.department\_id=d.department\_id

4 AND d.location\_id=l.location\_id

5\* AND c.country\_id=l.country\_id

1 SELECT e.first\_name,e.last\_name,e.salary, d.department\_name,l.city,c.country\_name,r.region\_name

2 FROM employees e, departments d,locations l,countries c,regions r

3 WHERE e.department\_id=d.department\_id

4 AND d.location\_id=l.location\_id

5 AND c.country\_id=l.country\_id

6\* AND r.region\_id=c.region\_id

1 SELECT e.first\_name,e.last\_name,e.salary, d.department\_name,l.city,c.country\_name,r.region\_name,jh.start\_date

2 FROM employees e, departments d,locations l,countries c,regions r,job\_history jh

3 WHERE e.department\_id=d.department\_id

4 AND d.location\_id=l.location\_id

5 AND c.country\_id=l.country\_id

6 AND r.region\_id=c.region\_id

7\* AND e.employee\_id=jh.employee\_id

1 SELECT e.first\_name,e.last\_name,e.salary, d.department\_name,l.city,c.country\_name,r.region\_name,jh.start\_date,j.job\_title

2 FROM employees e, departments d,locations l,countries c,regions r,job\_history jh,jobs j

3 WHERE e.department\_id=d.department\_id

4 AND d.location\_id=l.location\_id

5 AND c.country\_id=l.country\_id

6 AND r.region\_id=c.region\_id

7 AND e.employee\_id=jh.employee\_id

8\* AND j.job\_id=jh.job\_id

SQL> @c:/sqlqueries.txt

1 select first\_name,last\_name,job\_id,salary,(salary+salary\*.10) increased\_salary from employees

2\* WHERE LOWER(job\_id)='it\_prog'

SQL> DESCRIBE employees;

SQL> EXECUTE raise\_employee\_salary (206,10);

SQL> set serveroutput on;

SQL> EXECUTE raise\_employee\_salary (100,20);

SQL> SELECT MAX(salary),MIN(salary),AVG(salary),SUM(salary)

2 FROM employees;

SQL> SELECT MAX(salary),MIN(salary),AVG(salary),SUM(salary),tax(salary)

2 FROM employees;

1 CREATE OR REPLACE FUNCTION VA\_INCOMETAX

2 (salary IN NUMBER)

3 RETURN NUMBER

4 IS

5 BEGIN

6 RETURN (salary\*.05);

7\* END VA\_INCOMETAX;

1 SELECT first\_name,last\_name,salary,va\_incometax(salary)

2\* FROM employees

1 SELECT first\_name,last\_name,salary,va\_incometax(salary),(salary-va\_incometax(salary)) salary\_after\_tax

2\* FROM employees

1 CREATE TABLE employees\_triggertable1

2 AS

3 SELECT \* FROM employees

4\* WHERE employee\_id IN (100,101,102,103,104,105)

SQL> select \* from employees\_triggertable1;

1 CREATE TABLE employees\_triggertable2

2 AS

3\* select \* from employees\_triggertable1

SQL> select \* from employees\_triggertable2;

1 UPDATE employees\_triggertable1

2 SET first\_name='SSTTEEVVEENN'

3\* WHERE employee\_id=100

SQL> select \* from employees\_triggertable1;

SQL> select \* from employees\_triggertable2;

1\* select \* from employees\_triggertable2

1 update employees\_triggertable1

2 SET first\_name='STEVEN'

3\* where EMPLOYEE\_ID=100

SQL> select \* from employees\_triggertable1;

SQL> select \* from employees\_triggertable2;

1 update employees\_triggertable1

2 SET first\_name='Steven'

3\* WHERE employee\_id=100

SQL> select \* from employees\_triggertable1;

SQL> select \* from employees\_triggertable2;

CREATE OR REPLACE TRIGGER update\_trigger

BEFORE UPDATE

ON employees\_triggertable1

FOR EACH ROW

BEGIN

UPDATE employees\_triggertable2

SET employee\_id=:new.employee\_id,

first\_name =:new.first\_name,

last\_name =:new.last\_name,

email =:new.email,

phone\_number =:new.phone\_number,

hire\_date =:new.hire\_date,

job\_id =:new.job\_id,

salary =:new.salary,

commission\_pct =:new.commission\_pct,

manager\_id =:new.manager\_id,

department\_id =:new.department\_id

WHERE employee\_id=:old.employee\_id;

END;

/

SQL> select \* from employees\_triggertable1;

SQL> select \* from employees\_triggertable1;

UPDATE employees\_triggertable1

SET first\_name='SSSTTTEEEVVVEEENNN'

WHERE employee\_id=100

SQL> select \* from employees\_triggertable1;

SQL> select \* from employees\_triggertable2;

UPDATE employees\_triggertable2

SET first\_name='Steven'

WHERE employee\_id=100

SQL> select \* from employees\_triggertable1;

SQL> select \* from employees\_triggertable2;

CREATE OR REPLACE TRIGGER delete\_triggernew

BEFORE DELETE

ON employees\_triggertable3

FOR EACH ROW

BEGIN

DELETE FROM employees\_triggertable4

WHERE employee\_id=:old.employee\_id;

END;

SQL> select \* from employees\_triggertable3;

SQL> select \* from employees\_triggertable4;

DELETE from employees\_triggertable3

where employee\_id=100

SQL> select \* from employees\_triggertable3;

SQL> select \* from employees\_triggertable4;

select trigger\_name from user\_triggers

CREATE OR REPLACE FUNCTION get\_sal

2 (id NUMBER) RETURN NUMBER IS

3 sal NUMBER(8,2) :=0 ;

4 BEGIN

5 SELECT salary INTO sal FROM employees

6 WHERE employee\_id=id;

7 RETURN sal;

8\* END get\_sal;

1 SELECT \* FROM employees

2\* WHERE salary=get\_sal(100

1 SELECT first\_name,salary FROM employees

2\* WHERE salary=get\_sal(100)

1 CREATE OR REPLACE FUNCTION get\_sal1

2 (id employees.employee\_id%TYPE) RETURN NUMBER IS

3 sal employees.salary%TYPE :=0 ;

4 BEGIN

5 SELECT salary INTO sal FROM employees

6 WHERE employee\_id=id;

7 RETURN sal;

8\* END get\_sal1;

1 SELECT \* FROM employees

2\* WHERE salary=get\_sal1(100)

1 CREATE OR REPLACE TRIGGER update\_trigger1

2 BEFORE UPDATE

3 ON employees\_triggertable2

4 FOR EACH ROW

5 BEGIN

6 UPDATE employees\_triggertable1

7 SET employee\_id=:new.employee\_id,

8 first\_name =:new.first\_name,

9 last\_name =:new.last\_name,

10 email =:new.email,

11 phone\_number =:new.phone\_number,

12 hire\_date =:new.hire\_date,

13 job\_id =:new.job\_id,

14 salary =:new.salary,

15 commission\_pct =:new.commission\_pct,

16 manager\_id =:new.manager\_id,

17 department\_id =:new.department\_id

18 WHERE employee\_id=:old.employee\_id;

19\* END;

SQL> select \* from employees\_triggertable1;

SQL> select \* from employees\_triggertable2;

1 CREATE TABLE employees\_triggertable3

2 AS

3 SELECT \* FROM employees

4\* WHERE employee\_id IN (100,101,102,103,104,105)

5 /

1 CREATE TABLE employees\_triggertable4

2 AS

3 SELECT \* FROM employees

4\* WHERE employee\_id IN (100,101,102,103,104,105)

SQL> select \* from employees\_triggertable3;

SQL> select \* from employees\_triggertable4;

1 CREATE OR REPLACE TRIGGER update\_triggernew

2 BEFORE UPDATE

3 ON employees\_triggertable3

4 FOR EACH ROW

5 BEGIN

6 UPDATE employees\_triggertable4

7 SET employee\_id=:new.employee\_id,

8 first\_name =:new.first\_name,

9 last\_name =:new.last\_name,

10 email =:new.email,

11 phone\_number =:new.phone\_number,

12 hire\_date =:new.hire\_date,

13 job\_id =:new.job\_id,

14 salary =:new.salary,

15 commission\_pct =:new.commission\_pct,

16 manager\_id =:new.manager\_id,

17 department\_id =:new.department\_id

18 WHERE employee\_id=:old.employee\_id;

19\* END;

1 UPDATE employees\_triggertable3

2 SET first\_name='SSSTTTEEEVVVEEENNN'

3\* WHERE employee\_id=100

SQL> select \* from employees\_triggertable3;

SQL> select \* from employees\_triggertable4;

1 CREATE TRIGGER insert\_triggernew

2 AFTER INSERT

3 ON employees\_triggertable3

4 FOR EACH ROW

5 BEGIN

6 INSERT INTO employees\_triggertable4

7 VALUES (:new.employee\_id,:new.first\_name,:new.last\_name,:new.email,:new.phone\_number,:new.hire\_date, :new.job\_id,:new.salary,:new.commission\_pct,:new.manager\_id,:new.department\_id);

8\* END;

9 /

1 INSERT INTO employees\_triggertable3 (employee\_id,first\_name,last\_name,email,phone\_number,hire\_date,

2 job\_id,salary,commission\_pct,manager\_id,department\_id)

3\* VALUES (106,'Priya','Shetty','pshetty','703.235.2653','20-JUL-06','AC\_MGR',12000,.10,101,110)

SQL> /

SQL> select \* from employees\_triggertable3;

SQL> select \* from employees\_triggertable4;

SQL> set pagesize 300

SQL> set linesize 300

SQL> SELECT \* FROM employees;

SQL> SELECT \* FROM employees\_pinkey;

SQL> DESCRIBE employees\_pinkey;

SQL> SELECT table\_name FROM user\_tables;

1 CREATE TABLE employees\_pinkey

2 AS

3\* SELECT \* FROM employees

SQL> SELECT \* FROM employees\_pinkey;

SQL> DESCRIBE employees\_pinkey;

SQL> SELECT table\_name FROM user\_tables;

1 CREATE TABLE employees\_GTAS

2 AS

3\* SELECT \* FROM employees WHERE salary>AVG(salary)

1 CREATE TABLE employees\_GTAS

2 AS

3\* SELECT \* FROM employees WHERE salary> SELECT AVG(salary)FROM employees

1 CREATE TABLE employees\_GTAS

2 AS

3\* SELECT \* FROM employees WHERE salary> (SELECT AVG(salary)FROM employees)

SQL> SELECT \* FROM employees\_GTAS;

SQL> DESCRIBE employees\_GTAS;

SQL> SELECT table\_name FROM user\_tables;

SQL> CREATE TABLE employees\_esra

2 AS

3 SELECT \* FROM employees

4 WHERE 1=2;

SQL> SELECT \* FROM employees\_ESRA;

SQL> DESCRIBE employees\_ESRA;

SQL> SELECT table\_name FROM user\_tables;

1 INSERT INTO employees\_esra

2 (employee\_id,first\_name,last\_name,email,phone\_number,hire\_date,job\_id,salary,commission\_pct,manager\_id,department\_id)

3 VALUES

4\* (207,SHAMSUR,CHOWDHURY,schowdhury,718.345.4444,12-MAY-00,ACCOUNTANT,25001,.2,206,90)

1 INSERT INTO employees\_esra

2 (employee\_id,first\_name,last\_name,email,phone\_number,hire\_date,job\_id,salary,commission\_pct,manager\_id,department\_id)

3 VALUES

4\* (207,'SHAMSUR','CHOWDHURY','schowdhury','718.345.4444','12-MAY-00','ACCOUNTANT',25001,.2,206,90)

SQL> SELECT \* FROM employees\_esra;

SQL> @c:\insert01172009.txt

ERROR at line 4:

ORA-12899: value too large for column "HR"."EMPLOYEES\_ESRA"."JOB\_ID" (actual: 11, maximum: 10)

SQL> DESCRIBE employees\_esra;

SQL> ALTER TABLE employees\_esra

2 MODIFY (job\_id VARCHAR2(20));

SQL> DESCRIBE employees\_esra;

SQL> @c:\insert01172009.txt;

SQL> SELECT \* FROM employees\_esra;

SQL> ALTER TABLE employees\_esra

2 ADD (AGE NUMBER(3));

SQL> SELECT \* FROM employees\_esra;

SQL> @c:\updatetable01172009.txt

SQL> SELECT \* FROM employees\_esra;

SQL> Create Table employees\_gias

2 AS SELECT \* FROM employees\_esra;

Table created.

SQL> Create Table employees\_farida

2 AS SELECT \* FROM employees\_esra;

Table created.

SQL> SELECT \* FROM employees\_esra;

SQL> SELECT \* FROM employees\_gias;

SQL> DELETE FROM employees\_esra

2 WHERE employee\_id=211;

SQL> SELECT \* FROM employees\_esra;

SQL> DELETE FROM employees\_esra;

4 rows deleted.

SQL> SELECT \* FROM employees\_esra;

no rows selected

SQL> rollback;

Rollback complete.

SQL> SELECT \* FROM employees\_esra;

SQL> DELETE FROM employees\_esra;

5 rows deleted.

SQL> SELECT \* FROM employees\_esra;

no rows selected

SQL> describe employees\_esra;

SQL> rollback;

Rollback complete.

SQL> SELECT \* FROM employees\_esra;

SQL> SELECT \* FROM employees\_gias;

SQL> TRUNCATE TABLE employees\_farida

2 WHERE employee\_id=211

SQL> TRUNCATE TABLE employees\_farida;

Table truncated.

SQL> ROLLBACK;

Rollback complete.

SQL> SELECT \* FROM employees\_farida;

no rows selected

SQL> describe employees\_farida;

SQL> SELECT \* FROM employees\_gias;

SQL> FLASHBACK TABLE employees\_gias TO BEFORE DROP;

SQL> SELECT \* FROM employees\_gias;

SQL> select count(\*) FROM EMPLOYEES\_ESRA;

COUNT(\*)

----------

5

SQL> SELECT \* FROM employees\_esra;

SQL> DELETE FROM employees\_esra

2 WHERE employee\_id=211;

1 row deleted.

SQL> COMMIT;

Commit complete.

SQL> select count(\*) FROM EMPLOYEES\_ESRA;

COUNT(\*)

----------

4

SQL> select \* FROM employees;

SQL> SELECT employee\_id,first\_name,manager\_id

2 FROM employees;

SQL> select \* FROM employees;

SQL> select \* from departments;

SQL>

1 SELECT e.first\_name Employee, m.first\_name Manager

2 FROM employees e, employees m

3\* WHERE m.manager\_id=e.manager\_id

SQL>

1 SELECT e.first\_name Employee, m.first\_name Manager

2 FROM employees e, employees m

3\* WHERE e.manager\_id=m.employee\_id

1 SELECT first\_name, job\_id,salary,

2 DECODE(JOB\_ID, 'IT\_PROG', salary\*1.10,

3 'ST\_CLERK',salary\*1.15,

4 'SA\_MAN', salary\*1.20,

5 'SA\_REP', salary\*1.25,

6 salary) Final\_SALARY

7\* FROM employees

8 /

1 CREATE OR REPLACE FUNCTION NY\_STATE\_INCOMETAX(salary IN NUMBER)

2 RETURN NUMBER IS

3 BEGIN

4 RETURN (salary\*.08625);

5\* END NY\_STATE\_INCOMETAX;

SQL> select first\_name,salary, NY\_STATE\_INCOMETAX(salary) FROM employees;

select first\_name,salary, NY\_STATE\_INCOMETAX(salary) FROM employees

1 CREATE OR REPLACE PROCEDURE raise\_individual\_salary

2 (id NUMBER,

3 raisedpercent NUMBER)

4 IS

5 BEGIN

6 UPDATE employees

7 SET salary=salary+ (salary \* raisedpercent/100)

8 WHERE employee\_id=id;

9\* END raise\_individual\_salary;

10 /

SQL> SELECT \* FROM employees;

SQL> commit;

SQL> EXECUTE raise\_individual\_salary(206,10);

1 CREATE TABLE top\_emp\_manager1

2 AS

3 (SELECT m.first\_name, m.employee\_id, COUNT(e.employee\_id) total\_employee

4 FROM employees e, employees m

5 WHERE e.employee\_id=m.employee\_id

6\* GROUP BY m.first\_name, m.employee\_id)

1 SELECT total\_employee

2 FROM top\_emp\_manager1

3\* WHERE total\_employee=(SELECT MAX(total\_employee) FROM top\_emp\_manager1

1 SELECT \*

2\* FROM top\_emp\_manager1

3 ;

1 CREATE TABLE top\_emp\_manager

2 AS

3 (SELECT m.first\_name, m.employee\_id, COUNT(e.employee\_id) total\_employee

4 FROM employees e, employees m

5 WHERE e.manager\_id=m.employee\_id

6\* GROUP BY m.first\_name, m.employee\_id)

7 /

1 SELECT first\_name, employee\_id,total\_employee

2 FROM top\_emp\_manager

3\* WHERE total\_employee=(SELECT MAX(total\_employee) FROM top\_emp\_manager)

1 CREATE TABLE top\_emp\_manager

2 AS

3 (SELECT m.first\_name manager,COUNT(e.employee\_id) total\_employee

4 FROM employees e, employees m

5 WHERE e.manager\_id=m.employee\_id

6\* GROUP BY m.first\_name)

SQL> SELECT \* FROM top\_emp\_manager;

SQL> SELECT manager,total\_employee

2 FROM top\_emp\_manager

3 WHERE total\_employee=(SELECT MAX(total\_employee) FROM top\_emp\_manager);

SQL> SELECT manager,total\_employee

2 FROM top\_emp\_manager

3 WHERE total\_employee=(SELECT MIN(total\_employee) FROM top\_emp\_manager);

2 FROM top\_emp\_manager

3 WHERE total\_employee IN (SELECT MIN(total\_employee) FROM top\_emp\_manager);

1 CREATE TABLE top\_emp\_manager

2 AS

3 (SELECT m.first\_name manager,e.first\_name employee, COUNT(e.employee\_id) total\_employee

4 FROM employees e, employees m

5 WHERE e.manager\_id=m.employee\_id

6\* GROUP BY m.first\_name,e.first\_name)

SQL> SELECT \* FROM top\_emp\_manager;

1 SELECT manager,employee

2 FROM top\_emp\_manager

3\* WHERE UPPER(manager)='STEVEN'

select COUNT(manager) FROM top\_emp\_manager

1 select COUNT(manager) FROM top\_emp\_manager

2\* WHERE upper(manager)='STEVEN'

1 SELECT first\_name, last\_name, job\_id, salary,

2 DECODE (job\_id, 'IT\_PROG', salary\*1.10,

3 'FI\_ACCOUNT', salary\*1.15,

4 'ST\_CLERK', salary\*1.20,

5 'SA\_REP', salary\*1.25,

6 'AC\_ACCOUNT', salary\*1.30,

7 salary)

8\* FROM employees

1 SELECT first\_name, last\_name, job\_id, salary,

2 DECODE (job\_id, 'IT\_PROG', salary\*1.10,

3 'FI\_ACCOUNT', salary\*1.15,

4 'ST\_CLERK', salary\*1.20,

5 'SA\_REP', salary\*1.25,

6 'AC\_ACCOUNT', salary\*1.30,

7 salary) Increased\_Salary

8\* FROM employees

1 SELECT first\_name, last\_name, job\_id, salary

2 FROM employees

3\* WHERE job\_id IN ('IT\_PROG','FI\_ACCOUNT','ST\_CLERK','SA\_REP','AC\_ACCOUNT')

1 SELECT first\_name, last\_name, job\_id, salary,

2 DECODE (job\_id, 'IT\_PROG', salary\*1.10,

3 'FI\_ACCOUNT', salary\*1.15,

4 'ST\_CLERK', salary\*1.20,

5 'SA\_REP', salary\*1.25,

6 'AC\_ACCOUNT', salary\*1.30,

7 salary) increased\_salary

8\* FROM employees

1 SELECT first\_name, last\_name, job\_id, salary

2 FROM employees

3\* WHERE job\_id IN ('IT\_PROG','FI\_ACCOUNT','ST\_CLERK','SA\_REP','AC\_ACCOUNT')

1 UPDATE employees

2 SET salary=salary\*1.10

3\* WHERE employee\_id=103

SQL> select \* from employees;

1 select \* from employees

2\* order by DEPARTMENT\_ID

1 SELECT COUNT(department\_id)

2\* FROM employees

1 SELECT COUNT(DISTINCT(department\_id))

2\* FROM employees

1 SELECT DISTINCT(department\_id)

2 FROM employees

3\* WHERE department\_id IS NOT NULL

1 SELECT \*

2 FROM employees

3\* WHERE rownum<=10

1 select \* from employees

2\* order by employee\_id DESC

1 SELECT \*

2 FROM (select \* from employees

3 order by rownum DESC)

4\* WHERE rownum<=10

1 SELECT \*

2 FROM (select \* from employees

3 order by rownum DESC)

4 WHERE rownum<=10

5\* order by department\_id DESC

1 SELECT \*

2 FROM (select \* from employees

3 order by rownum DESC)

4 WHERE rownum<=10

5\* order by department\_id

1 SELECT \*

2 FROM (select \* from employees

3 order by rownum DESC)

4 WHERE rownum<=10

5\* order by employee\_id

1 SELECT first\_name, last\_name, job\_id, salary, increased\_salary

2 FROM (SELECT first\_name, last\_name, job\_id, salary,

3 DECODE (job\_id, 'IT\_PROG', salary\*1.10,

4 'FI\_ACCOUNT', salary\*1.15,

5 'ST\_CLERK', salary\*1.20,

6 'SA\_REP', salary\*1.25,

7 'AC\_ACCOUNT', salary\*1.30,

8 salary) increased\_salary

9 FROM employees)

10\* WHERE job\_id IN ('IT\_PROG','FI\_ACCOUNT','ST\_CLERK','SA\_REP','AC\_ACCOUNT')

11 /

SQL> select e.first\_name, d.department\_name, L.city, c.country\_name

2 from locations L, employees e, departments d, countries c

3 where e.department\_id = d.department\_id

4 and d.location\_id = L.location\_id

5 and L.country\_id = c.country\_id;

1 select e.first\_name, d.department\_name, L.city, c.country\_name

2 from locations L, employees e, departments d, countries c

3 where e.department\_id = d.department\_id

4 and d.location\_id = L.location\_id

5\* and L.country\_id = c.country\_id

1 select e.first\_name, e.Last\_name, e.salary, d.department\_name, L.city, r.region\_name, j.job\_title, jh.start\_date

2 from employees e, jobs j, job\_history jh, departments d, locations L, countries c, regions r

3 where e.department\_id = d.department\_id

4 and d.location\_id = l.location\_id

5 and l.country\_id = c.country\_id

6 and c.region\_id = r.region\_id

7 and d.department\_id = jh.department\_id

8\* and jh.job\_id = j.job\_id

9 /

1 Select DISTINCT(e.first\_name), e.Last\_name, e.salary, d.department\_name, L.city, r.region\_name, j.job\_title, jh.start\_date

2 from employees e, jobs j, job\_history jh, departments d, locations L, countries c, regions r

3 where e.department\_id = d.department\_id

4 and d.location\_id = l.location\_id

5 and l.country\_id = c.country\_id

6 and c.region\_id = r.region\_id

7 and d.department\_id = jh.department\_id

8\* and jh.job\_id = j.job\_id

1 Select DISTINCT e.first\_name, e.Last\_name, e.salary, d.department\_name, L.city, r.region\_name, j.job\_title, jh.start\_date

2 from employees e, jobs j, job\_history jh, departments d, locations L, countries c, regions r

3 where e.department\_id = d.department\_id

4 and d.location\_id = l.location\_id

5 and l.country\_id = c.country\_id

6 and c.region\_id = r.region\_id

7 and d.department\_id = jh.department\_id

8\* and jh.job\_id = j.job\_id

1 Select e.First\_name, e.Last\_name, e.salary, d.department\_name, JH.Start\_date, J.job\_title, L.city, R.region\_name

2 From employees e, departments d, job\_history JH, jobs J, locations L, regions R, Countries C

3 WHERE e.department\_id = d.department\_id

4 AND d.department\_id = JH.department\_id

5 AND d.location\_id = L.location\_id

6 AND L.country\_id = C.country\_id

7 AND C.region\_id = R.region\_id

8\* AND JH.Job\_id = j.job\_id

9 /

1 Select e.first\_name, e.last\_name, e.job\_id, r.region\_name

2 FROM employees e, departments d, locations l, countries c, regions r

3 WHERE e.department\_id = d.department\_id

4 AND d.location\_id = l.location\_id

5 AND l.country\_id = c.country\_id

6\* AND c.region\_id = r.region\_id

1 select \*from employees

2 where job\_id = 'st\_clerk'

3\* and hire\_date > '31-dec-1997'

1 select \* from employees

2 where job\_id = 'st\_clerk'

3\* and hire\_date > '31-dec-1997'

4 /

1 select \* from employees

2 where lower(job\_id) = 'st\_clerk'

3\* and hire\_date > '31-dec-1997'

4 /

1 select \* from employees

2 where job\_id = 'ST\_CLERK'

3\* and hire\_date > '31-dec-1997'

4 /

SQL> select last\_name, job\_id, salary, commission\_pct

2 from employees

3 where commission\_pct is NOT NULL

4 ORDER BY salary DESc;

SQL> select first\_name, last\_name, salary, commission\_pct, round(salary\*1.10,0)

2 from employees

3 where commission\_pct IS NULL;

SQL> select 'The salary of '||last\_name||' after a 10% raise is '||ROUND(salary\*1.10)"New Salary"

2 from employees

3 where commission\_pct IS NULL;

1 SELECT last\_name,MONTHS\_BETWEEN(SYSDATE,hire\_date) Total\_months, MONTHS\_BETWEEN(SYSDATE,hire\_date)/12 Total\_years

2\* FROM employees

1 SELECT last\_name,TRUNC(MONTHS\_BETWEEN(SYSDATE,hire\_date)) Total\_months, (MONTHS\_BETWEEN(SYSDATE,hire\_date)/12) Total\_years

2\* FROM employees

1 SELECT last\_name,TRUNC(MONTHS\_BETWEEN(SYSDATE,hire\_date)) Total\_months, TRUNC(MONTHS\_BETWEEN(SYSDATE,hire\_date)/12) Total\_years

2\* FROM employees

1 SELECT last\_name,TRUNC(MONTHS\_BETWEEN(SYSDATE,hire\_date)/12) years, TRUNC(MOD(MONTHS\_BETWEEN(SYSDATE,hire\_date),12)) months

2\* FROM employees

1 SELECT last\_name|| 'has been working '||TRUNC(MONTHS\_BETWEEN(SYSDATE,hire\_date)/12) ||' years and '|| TRUNC(MOD(MONTHS\_BETWEEN(SYSDATE,hire\_date),12))|| ' months'

2\* FROM employees

1 SELECT last\_name|| ' has been working '||TRUNC(MONTHS\_BETWEEN(SYSDATE,hire\_date)/12) ||' years and '|| TRUNC(MOD(MONTHS\_BETWEEN(SYSDATE,hire\_date),12))|| ' months.'

2\* FROM employees

1 SELECT RPAD(last\_name,11)|| ' has been working '||TRUNC(MONTHS\_BETWEEN(SYSDATE,hire\_date)/12) ||' years and '|| TRUNC(MOD(MONTHS\_BETWEEN(SYSDATE,hire\_date),12))|| ' months.'

2\* FROM employees

1 SELECT RPAD(last\_name,11,'\*')|| ' has been working '||TRUNC(MONTHS\_BETWEEN(SYSDATE,hire\_date)/12) ||' years and '|| TRUNC(MOD(MONTHS\_BETWEEN(SYSDATE,hire\_date),12))|| ' months.'

2\* FROM employees

1 SELECT RPAD(last\_name,11,'.')|| ' has been working '||TRUNC(MONTHS\_BETWEEN(SYSDATE,hire\_date)/12) ||' years and '|| TRUNC(MOD(MONTHS\_BETWEEN(SYSDATE,hire\_date),12))|| ' months.'

2\* FROM employees

1 SELECT RPAD(last\_name,11,'.')|| ' has been working '||LPAD(TRUNC(MONTHS\_BETWEEN(SYSDATE,hire\_date)/12),'0') ||' years and '|| TRUNC(MOD(MONTHS\_BETWEEN(SYSDATE,hire\_date),12))|| ' months.'

2\* FROM employees

1 SELECT RPAD(last\_name,11,'.')|| ' has been working '||LPAD(TRUNC(MONTHS\_BETWEEN(SYSDATE,hire\_date)/12),2,'0') ||' years and '|| LPAD(TRUNC(MOD(MONTHS\_BETWEEN(SYSDATE,hire\_date),12)),2,'0')|| ' months.'

2\* FROM employees

1 SELECT RPAD(last\_name,11)|| ' has been working '||LPAD(TRUNC(MONTHS\_BETWEEN(SYSDATE,hire\_date)/12),2,'0') ||' years and '|| LPAD(TRUNC(MOD(MONTHS\_BETWEEN(SYSDATE,hire\_date),12)),2,'0')|| ' months.'

2\* FROM employees

1 SELECT last\_name, salary

2\* FROM employees

1 SELECT last\_name, salary

2 FROM employees

3\* WHERE last\_name LIKE 'K%'

1 SELECT last\_name, salary

2 FROM employees

3\* WHERE last\_name LIKE 'J%'

4 /

1 SELECT last\_name, salary

2 FROM employees

3\* WHERE last\_name LIKE 'L%'

1 SELECT last\_name, salary

2 FROM employees

3\* WHERE last\_name LIKE 'M%'

1 (SELECT last\_name, salary

2 FROM employees

3 WHERE last\_name LIKE 'J%')

4 UNION

5 (SELECT last\_name, salary

6 FROM employees

7 WHERE last\_name LIKE 'K%')

8 UNION

9 (SELECT last\_name, salary

10 FROM employees

11 WHERE last\_name LIKE 'L%')

12 UNION

13 (SELECT last\_name, salary

14 FROM employees

15\* WHERE last\_name LIKE 'M%')

1 SELECT last\_name, salary

2 FROM employees

3\* WHERE last\_name LIKE 'J%' OR last\_name LIKE 'K%' OR last\_name LIKE 'L%' OR last\_name LIKE 'M%

1 SELECT last\_name, salary

2 FROM employees

3 WHERE last\_name LIKE 'J%' OR last\_name LIKE 'K%' OR last\_name LIKE 'L%' OR last\_name LIKE 'M%'

4\* ORDER BY last\_name

1 SELECT last\_name, salary

2 FROM employees

3 WHERE last\_name IN ('J%','K%', 'L%','M%')

4\* ORDER BY last\_name

1 SELECT last\_name, salary

2 FROM employees

3 WHERE last\_name LIKE IN ('J%','K%', 'L%','M%')

4\* ORDER BY last\_name

1\* SELECT SUBSTR(last\_name,1,2), salary FROM employees

1\* SELECT last\_name,SUBSTR(last\_name,1,2), salary FROM employees

1\* SELECT last\_name,SUBSTR(last\_name,1,3), salary FROM employees

1\* SELECT last\_name,SUBSTR(last\_name,1,1), salary FROM employees

1\* SELECT last\_name,SUBSTR(last\_name,1,1), SUBSTR(last\_name,2,1),salary FROM employees

1\* SELECT last\_name,SUBSTR(last\_name,2,1),salary FROM employees

1 SELECT last\_name,salary

2 FROM employees

3\* WHERE SUBSTR(last\_name,1,1)=('J','K','L','M')

1 SELECT last\_name,salary

2 FROM employees

3\* WHERE SUBSTR(last\_name,1,1) IN('J','K','L','M')

1 SELECT last\_name,salary

2 FROM employees

3 WHERE SUBSTR(last\_name,1,1) IN('J','K','L','M')

4\* ORDER BY last\_name

1 SELECT last\_name,salary

2 FROM employees

3 WHERE last\_name LIKE '%N'

4\* ORDER BY last\_name

1 SELECT last\_name,salary

2 FROM employees

3 WHERE UPPER(last\_name) LIKE '%N'

4\* ORDER BY last\_name

1 SELECT last\_name,salary

2 FROM employees

3 WHERE LOWER(last\_name) LIKE '%n'

4\* ORDER BY last\_name

1 SELECT last\_name,salary

2 FROM employees

3 WHERE LOWER(last\_name) LIKE '%n'

4\* ORDER BY last\_name

1 SELECT last\_name, SUBSTR(last\_name, -1), SUBSTR(last\_name, 1)

2\* FROM employees

1 SELECT last\_name, SUBSTR(last\_name, -1), SUBSTR(last\_name, 1,1)

2\* FROM employees

1 SELECT last\_name, salary

2 FROM employees

3\* WHERE UPPER(SUBSTR(last\_name, -1))='N'

1 SELECT e.first\_name, m.first\_name

2 FROM employees e, employees m

3\* WHERE e.manager\_id = m.employee\_id

4 /

1 SELECT e.first\_name employee, m.first\_name manager

2 FROM employees e, employees m

3\* WHERE e.manager\_id = m.employee\_id

1 SELECT e.first\_name employee, m.first\_name manager

2 FROM employees e, employees m

3\* WHERE e.manager\_id = m.employee\_id(+)

1 (SELECT e.first\_name employee, m.first\_name manager

2 FROM employees e, employees m

3 WHERE e.manager\_id = m.employee\_id(+))

4 MINUS

5 (SELECT e.first\_name employee, m.first\_name manager

6 FROM employees e, employees m

7\* WHERE e.manager\_id = m.employee\_id)

8 /

1 SELECT department\_id,AVG(salary) FROM employees

2\* GROUP BY department\_id

1 SELECT department\_id,AVG(salary) FROM employees

2 GROUP BY department\_id

3\* ORDER BY department\_id

1 SELECT department\_id,AVG(salary) FROM employees

2 WHERE department\_id IS NOT NULL

3 GROUP BY department\_id

4\* ORDER BY department\_id

1 SELECT department\_id,AVG(salary) FROM employees

2 WHERE department\_id IS NOT NULL

3 AND AVG(salary)>10000

4 GROUP BY department\_id

5\* ORDER BY department\_id

1 SELECT department\_id,AVG(salary) FROM employees

2 WHERE department\_id IS NOT NULL

3 GROUP BY department\_id

4 ORDER BY department\_id

5\* HAVING AVG(salary)>10000

1 SELECT department\_id,AVG(salary) FROM employees

2 WHERE department\_id IS NOT NULL

3 GROUP BY department\_id

4 HAVING AVG(salary)>10000

5\* ORDER BY department\_id

6 /

1 SELECT m.first\_name manager, m.employee\_id manager\_id, count(e.employee\_id)

2 FROM employees e, employees m

3 WHERE e.manager\_id = m.employee\_id

4 group by m.employee\_id, m.first\_name

5\* having count(e.employee\_id)=1

6 /

1 SELECT m.first\_name manager, m.employee\_id manager\_id, count(e.employee\_id)

2 FROM employees e, employees m

3 WHERE e.manager\_id = m.employee\_id

4 group by m.employee\_id, m.first\_name

5\* having count(e.employee\_id)>1

SQL> SELECT COUNT(\*) FROM employees;

1\* SELECT COUNT(employee\_id) FROM employees

1\* SELECT COUNT(department\_id) FROM employees

1\* SELECT COUNT(DISTINCT(department\_id)) FROM employees

1 SELECT COUNT(first\_name) FROM employees

2\* WHERE UPPER(first\_name)='STEVEN'

1 SELECT m.first\_name manager, m.employee\_id manager\_id, count(e.employee\_id)

2 FROM employees e, employees m

3 WHERE e.manager\_id = m.employee\_id

4 AND UPPER(m.first\_name)='STEVEN'

5 group by m.employee\_id, m.first\_name

6\* having count(e.employee\_id)>1

7 /

SQL> SELECT table\_name FROM user\_tables;

SQL> SELECT COUNT(\*) FROM user\_tables;

SQL> SELECT table\_name FROM tabs;

SQL> SELECT COUNT(\*) FROM tabs;

1 SELECT \* FROM employees

2\* ORDER BY first\_name

1 SELECT \* FROM employees

2 WHERE UPPER(first\_name)='PETER'

3\* OR UPPER(last\_name)='HALL'

1 SELECT \* FROM employees

2\* WHERE UPPER(first\_name) LIKE 'P%'

1\* SELECT salary,commission\_pct FROM employees

1\* SELECT first\_name,last\_name,salary,commission\_pct FROM employees

1 SELECT first\_name,last\_name,salary,commission\_pct, salary+salary\*commission\_pct FROM

2\* employees

3 /

1\* SELECT first\_name,last\_name,salary,commission\_pct, salary+salary\*NVL(commission\_pct,0) FROM employees

1\* SELECT first\_name,last\_name,salary,commission\_pct, salary+salary\*NVL(commission\_pct,.1) FROM employees

1 (SELECT last\_name FROM employees WHERE last\_name LIKE 'J%')

2 UNION

3 (SELECT last\_name FROM employees WHERE last\_name LIKE 'K%')

4 UNION

5 (SELECT last\_name FROM employees WHERE last\_name LIKE 'L%')

6 UNION

7\* (SELECT last\_name FROM employees WHERE last\_name LIKE 'M%')

1 (SELECT last\_name FROM employees WHERE last\_name LIKE 'J%')

2 UNION

3 (SELECT last\_name FROM employees WHERE last\_name LIKE 'K%')

4 UNION

5 (SELECT last\_name FROM employees WHERE last\_name LIKE 'L%')

6 UNION

7\* (SELECT last\_name FROM employees WHERE last\_name LIKE 'M%')

1 (SELECT last\_name FROM employees WHERE last\_name LIKE 'J%')

2 UNION ALL

3 (SELECT last\_name FROM employees WHERE last\_name LIKE 'K%')

4 UNION ALL

5 (SELECT last\_name FROM employees WHERE last\_name LIKE 'L%')

6 UNION ALL

7\* (SELECT last\_name FROM employees WHERE last\_name LIKE 'M%')

1 CREATE OR REPLACE FUNCTION incometax (salary IN number)

2 RETURN number IS

3 BEGIN

4 RETURN (salary\*.05);

5\* END incometax ;

1 select salary, incometax(salary)

2\* from employees

1 select last\_name,salary, incometax(salary)

2\* from employees

1 select last\_name,salary, incometax((salary+salary\*NVL(commission\_pct,0))\*12) YearlyIncomeTax

2\* from employees

1 select last\_name,salary, salary+salary\*.10 increased\_salary

2\* from employees

1 CREATE OR REPLACE PROCEDURE raise\_salary

2 (id NUMBER,

3 increased\_percent NUMBER)

4 IS

5 BEGIN

6 UPDATE employees

7 SET salary=salary+(salary\*increased\_percent)

8 WHERE employee\_id =id;

9\* END raise\_salary;

SQL> EXECUTE raise\_salary(101,10);

PL/SQL procedure successfully completed.

SQL> select \* from employees;

1 CREATE OR REPLACE PROCEDURE raise\_salary

2 (id NUMBER)

3 IS

4 BEGIN

5 UPDATE employees

6 SET salary=salary+(salary\*.1)

7 WHERE employee\_id =id;

8\* END raise\_salary;

SQL> EXECUTE raise\_salary(205);

PL/SQL procedure successfully completed.

SQL> select \* from employees;

1 CREATE OR REPLACE PROCEDURE raise\_individual\_salary

2 (id NUMBER,

3 raisedpercent NUMBER)

4 IS

5 BEGIN

6 UPDATE employees

7 SET salary=salary+ (salary \* raisedpercent/100)

8 WHERE employee\_id=id;

9\* END raise\_individual\_salary;

10 /

SQL> select \* from employees;

SQL> CREATE TABLE employees\_triggertable1

2 AS

3 SELECT \* FROM employees

4 WHERE employee\_id IN (100,101,102,103,104);

Table created.

SQL> CREATE TABLE employees\_triggertable2

2 AS

3 SELECT \* FROM employees\_triggertable1;

Table created.

SQL> ed

Wrote file afiedt.buf

1 CREATE TABLE employees\_triggertable2

2 AS

3\* SELECT \* FROM employees\_triggertable1

SQL> ed

Wrote file afiedt.buf

1 CREATE TABLE employees\_triggertable3

2 AS

3 SELECT \* FROM employees

4\* WHERE employee\_id BETWEEN 100 AND 104

5 /

Table created.

SQL> SELECT \* FROM employees\_triggertable1;

SQL> SELECT \* FROM employees\_triggertable2;

SQL> SELECT \* FROM employees\_triggertable3;

CREATE OR REPLACE TRIGGER update\_trigger

2 BEFORE UPDATE

3 ON employees\_triggertable1

4 FOR EACH ROW

5 BEGIN

6 UPDATE employees\_triggertable2

7 SET employee\_id=:new.employee\_id,

8 first\_name =:new.first\_name,

9 last\_name =:new.last\_name,

10 email =:new.email,

11 phone\_number =:new.phone\_number,

12 hire\_date =:new.hire\_date,

13 job\_id =:new.job\_id,

14 salary =:new.salary,

15 commission\_pct =:new.commission\_pct,

16 manager\_id =:new.manager\_id,

17 department\_id =:new.department\_id

18 WHERE employee\_id=:old.employee\_id;

19\* END;

1 UPDATE employees\_triggertable1

2 SET salary=salary+salary\*.10

3\* WHERE employee\_id=101

SQL> SELECT \* FROM employees\_triggertable1;

SQL> SELECT \* FROM employees\_triggertable2;

SQL> SELECT \* FROM employees\_triggertable3;

1 CREATE OR REPLACE TRIGGER delete\_trigger

2 BEFORE DELETE

3 ON employees\_triggertable1

4 FOR EACH ROW

5 BEGIN

6 DELETE FROM employees\_triggertable2

7 WHERE employee\_id=:old.employee\_id;

8\* END;

SQL> DELETE FROM employees\_triggertable1

2 WHERE employee\_id=104;

1 CREATE OR REPLACE FUNCTION get\_sal

2 (id NUMBER) RETURN NUMBER IS

3 sal NUMBER(8,2) :=0 ;

4 BEGIN

5 SELECT salary INTO sal FROM employees

6 WHERE employee\_id=id;

7 RETURN sal;

8\* END get\_sal;

SQL> SELECT \* FROM employees

2 WHERE salary=get\_sal(200);

1 CREATE OR REPLACE FUNCTION get\_sal1

2 (id employees.employee\_id%TYPE) RETURN NUMBER IS

3 sal employees.salary%TYPE :=0 ;

4 BEGIN

5 SELECT salary INTO sal FROM employees

6 WHERE employee\_id=id;

7 RETURN sal;

8\* END get\_sal1;

9 /

Function created.

SQL> EXECUTE dbms\_output.put\_line(get\_sal1(198));

1 CREATE TRIGGER insert\_trigger

2 AFTER INSERT

3 ON employees\_triggertable1

4 FOR EACH ROW

5 BEGIN

6 INSERT INTO employees\_triggertable2

7 VALUES (:new.employee\_id,:new.first\_name,:new.last\_name,:new.email,:new.phone\_number,:new.hire\_date, :new.job\_id,:new.salary,:new.commission\_pct,:new.manager\_id,:new.department\_id);

8\* END;

9 /

1 INSERT INTO employees\_triggertable1 (employee\_id,first\_name,last\_name,email,phone\_number,hire\_date,

2 job\_id,salary,commission\_pct,manager\_id,department\_id)

3\* VALUES (104,'Fari','Haque','fhaque','703.235.2653','20-JUL-06','AC\_MGR',12000,.10,101,110)

SQL> SELECT COUNT(department\_id) FROM employees;

COUNT(DEPARTMENT\_ID)

--------------------

106

SQL> SELECT COUNT(DISTINCT(department\_id)) FROM employees;

COUNT(DISTINCT(DEPARTMENT\_ID))

------------------------------

11

SQL> SELECT first\_name,department\_id FROM employees;

1 SELECT department\_id, COUNT(DISTINCT(department\_id)) FROM employees

2\* GROUP BY department\_id

SQL> SELECT department\_id, COUNT(department\_id) FROM employees

2 GROUP BY department\_id;

1 SELECT department\_id, COUNT(department\_id) FROM employees

2 GROUP BY department\_id

3\* ORDER BY department\_id

SQL> SELECT department\_id,AVG(salary)

2 FROM employees

3 GROUP BY department\_id;

SQL> SELECT department\_id,AVG(salary)

2 FROM employees

3 GROUP BY department\_id

4 ORDER BY department\_id;

1 SELECT department\_id,AVG(salary)

2 FROM employees

3 WHERE department\_id>100

4 GROUP BY department\_id

5 HAVING AVG(salary)>10000

6\* ORDER BY department\_id

SQL> rem creating a view from employees table using name and phone number.

SQL> CREATE VIEW customer\_view

2 AS

3 SELECT first\_name,last\_name,phone\_number

4 FROM employees;

View created.

SQL> commit;

Commit complete.

SQL> SELECT \* FROM customer\_view;

SQL> UPDATE customer\_view

2 SET phone\_number='516-234-4567'

3 WHERE UPPER(first\_name)='WILLIAM';

2 rows updated.

SQL> SELECT \* FROM customer\_view;

SQL> create synonym cview

2 FOR customer\_view;

Synonym created.

SQL> SELECT \* FROM cview;

SQL> SELECT \* FROM customer\_view;

1 SELECT \* FROM employees

2 WHERE rownum<11

3\* ORDER BY salary desc

SQL> CONNECT hr/hr

Connected.

SQL> DISCONNECT

1 SELECT \*

2 FROM employees

3\* WHERE UPPER(first\_name)=('STEVEN','NEENA','LEX')

1 SELECT \*

2 FROM employees

3\* WHERE UPPER(first\_name)=('STEVEN','NEENA','LEX')

1 SELECT \*

2 FROM employees

3\* WHERE UPPER(first\_name) IN ('STEVEN','NEENA','LEX')

1 (SELECT \*

2 FROM employees

3 WHERE UPPER(first\_name) LIKE 'N%')

4 UNION

5 (SELECT \*

6 FROM employees

7\* WHERE UPPER(first\_name) LIKE 'S%')

8 /

1 (SELECT \*

2 FROM employees

3 WHERE UPPER(first\_name) LIKE 'N%')

4 UNION

5 (SELECT \*

6 FROM employees

7 WHERE UPPER(first\_name) LIKE 'S%')

8\* ORDER BY 1

1 (SELECT \*

2 FROM employees

3 WHERE UPPER(first\_name) LIKE 'N%')

4 UNION

5 (SELECT \*

6 FROM employees

7 WHERE UPPER(first\_name) LIKE 'S%')

8\* ORDER BY 2

1 ((SELECT \*

2 FROM employees

3 WHERE UPPER(first\_name) LIKE 'N%')

4 UNION

5 (SELECT \*

6 FROM employees

7 WHERE UPPER(first\_name) LIKE 'S%'))

8\* ORDER BY 'first\_name'

SQL> SELECT INSTR('HANIP','N') FROM dual;

INSTR('HANIP','N')

------------------

3

SQL> ed

Wrote file afiedt.buf

1\* SELECT INSTR('HANIP','P') FROM dual

SQL> /

INSTR('HANIP','P')

------------------

5

1 SELECT \*

2 FROM employees

3\* WHERE UPPER(first\_name) IN ('S','N','P')

SQL> SELECT INSTR('HANIP','N') FROM dual;

INSTR('HANIP','N')

------------------

3

SQL> SELECT SUBSTR('HANIP',1,3) FROM dual;

SUB

---

HAN

SQL> SELECT SUBSTR('HANIP',2,3) FROM dual;

SUB

---

ANI

SQL> SELECT SUBSTR('HANIP',1,1) FROM dual;

S

-

H

SQL> ED

Wrote file afiedt.buf

1 SELECT \*

2 FROM employees

3\* WHERE SUBSTR(UPPER(first\_name),1,1) IN ('S','N','P')

4

1 (SELECT \*

2 FROM employees

3 WHERE UPPER(first\_name) LIKE 'S%')

4 UNION

5 (SELECT \*

6 FROM employees

7 WHERE UPPER(first\_name) LIKE 'N%')

8 UNION

9 (SELECT \*

10 FROM employees

11\* WHERE UPPER(first\_name) LIKE 'P%')

12 /