# *Database Management I (420-D10-HR)*

# *Lab 15 – Views, Sequences, Indicies*

Date assigned: Monday, November 21, 2016

Date due: **Monday, November 21, 2016**

**Learning Objectives**

Upon successful completion of this lab exercise, the student will be able to:

* 1. Create and use a view;
  2. Create and use a sequence;
  3. Create and use an index;

***To uploaded to Moodle:***

1. The ***username*\_D10\_L15\_Advanced\_SQL** folder should be zipped and uploaded to **Moodle**.

***To Start:***

1. Rename this document ***username*\_D10\_L15\_Advanced\_SQL** and submit it into Moodle with your answers.
2. Read the following links:
   1. [View Tutorial](https://www.techonthenet.com/oracle/views.php), more [advanced Views](https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_8004.htm)
   2. [Indicies](https://www.techonthenet.com/oracle/indexes.php)
   3. [Sequences](https://www.techonthenet.com/oracle/sequences.php)

**Marks:**

|  |  |  |
| --- | --- | --- |
| **Question** | **Mark** | **Out of** |
| Part A –Views, Sequences, Indicies |  |  |
| 1 |  | 5 |
| 2 |  | 6 |
| 3 |  | 6 |
| 4 |  | 2 |
| 5 |  | 6 |
| 6 |  | 8 |
| Part B – SQL Developer ; Table Structure |  | 0 |
| 3 |  | 2 |
| Part C – SQL Developer ; Table Data |  | 0 |
| 3 |  | 2 |
| Organization |  | 2 |
| **Total** |  | **31** |

# Views, Sequences, Indicies

**Objectives:** Learn to use SQL to create a view, an index and a sequence.

Learn to add a row to a table using a sequence for the primary key.

Use the NN database and IU database provided as part of the Shah DB package. If you need a fresh copy of this, you can find it in Moodle under “Databases for labs”

***To Do:***

## Create a view to include department name and average salary by department. Call your view **nn\_dept\_salaries\_vw**.

Show me your SQL and a sample of your view output (select \* from …)

DROP VIEW nn\_dept\_salaries\_vw;

CREATE VIEW nn\_dept\_salaries\_vw AS (

SELECT DISTINCT d.deptname, AVG(e.salary) AS "Average Salary"

FROM nn\_dept d, nn\_employee e

WHERE e.deptid = d.deptid

group by d.deptname

);

SELECT \*

FROM nn\_dept\_salaries\_vw;

## Write a query that uses the **nn\_dept\_salaries\_vw** view to list all the employees whose salary is less than the average salary for their department. Your output should look like the following:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Employee id** | **Name** | **Employee**  **Salary** | **Department** | **Department**  **Average Salary** |
| 433 | McCall, Alex | $66,500 | InfoSys | $73,250 |
| 222 | Chen, Sunny | $35,000 | Finance | $125,000 |
| 123 | Roberts, Sandi | $75,000 | Finance | $125,000 |
| 200 | Shaw, Jinku | $24,500 | Sales | $34,750 |

SQL and sample output:

SELECT e.employeeid, e.lname||' '||e.fname AS "Name",

e.salary, d.deptname, "Average Salary"

FROM nn\_dept d, nn\_employee e, nn\_dept\_salaries\_vw s

WHERE e.salary < s."Average Salary"

AND d.deptid = e.deptid;

## Create a view that will display name, department number and total income (salary + commission) of each employee in Department 10. Prevent a change of department through the view. Call your view **nn\_dept\_10\_vw**. The view should contain the following:

|  |  |  |
| --- | --- | --- |
| **NAME** | **DEPTID** | **INCOME** |
| John Smith | 10 | 300000 |
| Sandi Roberts | 10 | 75000 |
| Sunny Chen | 10 | 35000 |

SQL and sample output:

DROP VIEW nn\_dept\_10\_vw;

CREATE VIEW nn\_dept\_10\_vw AS (

SELECT DISTINCT e.lname||', '||e.lname AS "Name",

e.deptid, e.salary + NVL(e.commission, 0) AS "Income"

FROM nn\_employee e

WHERE deptid = 10

) WITH READ ONLY;

SELECT \*

FROM nn\_dept\_10\_vw;

## Create an index to search students faster based on their Major ID. Call your index **iu\_student\_majorid\_idx**. Answer the following: What is the benefit of an index?

SQL to create your index:

DROP INDEX iu\_student\_majorid\_idx;

CREATE INDEX iu\_student\_majorid\_idx

ON iu\_student (majorid);

## Create a sequence to add room IDs, and then insert a new room into the LOCATION table using the newly created sequence. What is the CURRVAL after the new row is inserted?. Call your sequence **iu\_location\_roomid\_seq**. It should start with 45. For the insert, add room 221 in the building 'Heritage'. It is a classroom and has a capacity of 40.

SQL and sample output:

DROP SEQUENCE iu\_location\_roomid\_seq;

CREATE SEQUENCE iu\_location\_roomid\_seq

Start With 45

increment by 1;

INSERT INTO iu\_location (

roomid, building, roomno, capacity, roomtype

)

values (

iu\_location\_roomid\_seq.NEXTVAL, 'Heritage', 221, 40 , 'C'

);

## Create a sequence deptid\_seq to generate department ID (in the DEPT table) and another sequence empid\_seq to generate employee Id (in the EMPLOYEE table). Use depid\_seq to add a new department in the DEPT table. Now, add yourself as a new employee with empid\_seq in the department you just added. Be sure to configure your sequences to start with values larger than in the current database.

SQL and sample outputs:

DROP SEQUENCE deptid\_seq;

DROP SEQUENCE empid\_seq;

CREATE SEQUENCE deptid\_seq

START WITH 50

INCREMENT BY 10;

CREATE SEQUENCE empid\_seq

START WITH 544

INCREMENT BY 1;

INSERT INTO nn\_dept (

deptid, deptname, location, employeeid

)

values (

deptid\_seq.nextval, 'Web dev', 'Ottawa', empid\_seq.nextval

);

INSERT INTO nn\_employee (

employeeid, lname, fname, positionid, supervisor,

hiredate, salary, commission, deptid, qualid

)

values (

empid\_seq.nextval, 'Dumaresq', 'Philip', 3, 111,

TO\_DATE('30-MAY-2017', 'DD-MON-RR'), 100000, null, deptid\_seq.currval, 1

);

# Working with a Table Structure in SQL Developer

***Purpose***: Learn to view and modify a table structure and to add constraints to an existing table using **SQL Developer**.

***To Do:***

## Open the **iu\_course** table.

## Add a new column by selecting the **Action** icon and then selecting **Column** 🡪 **Add**. The new column is:

DeptId – number(3,0), allows nulls

If you don't see your new column in the display, click the **Refresh** button.

## Click the **Constraints** tab. Click the edit symbol (The pencil and paper icon.) Add a foreign key constraint for **DeptId**. Call the constraint **IU\_COURSE\_DEPARTMENT\_FK**. It should reference the **iu\_department** table. It should cascade on delete.

Show me a screen capture of your new Constraint:

# Working with Table Data

***Purpose***: Learn to view and modify table data using **SQL Developer**.

***To Do:***

## Use the update statement to change the **DeptId** to 420 for the Heritage Computer Science courses (course number begins with **420**.) Commit your changes.

## Open the **iu\_courses** table and select the **Data** tab. Add a new row for course 420-D20 – Database Management II. It has 2 theory hours, 2 lab hours, 2 homework hours and is worth 2 credits. 420-D10 is a prerequisite for it. It belongs to department 420. **Commit** the changes.

## Now we want to see only those rows with 2 credits and in dept 420. Type *credits = 2 and deptid = 420* in the **Filter** text field and hit enter.

Show me a screen capture of this last step: