# *Database Management II (420-D20-HR)*

# *Happy Valley Kennels Project*

# *Assignment 1 - Introduction to PL/SQL*

Date assigned: Tuesday, January 31, 2016

Date due: **Wednesday, February 15, 2016, 11:50pm**

**Learning Objectives**

After completing this assignment, the student will be able to:

1. Write an anonymous PL/SQL block;
2. Declare and use PL/SQL variables;
3. Use the PL/SQL SELECT statement in a block;
4. Use SQL DML statements in a PL/SQL block;
5. Use an implicit cursor in a PL/SQL block.

**To be uploaded to Moodle:**

1. The***username*\_D20\_A01\_PL\_SQL\_Intro**.**docx** file containing the SQL source code and sample output for this assignment.

**To Start:**

1. Rebuild the HVK tables by finding the drop and create scripts in Moodle then running it in SQL Developer.
2. Rename this document to ***username*\_D20\_A01\_PL\_SQL\_Intro**.**docx** and insert your solutions within it.
3. You **must** provide effort feedback for this assignment or marks will be deducted (“handed in properly”)

**Marking and Time management:**

|  |  |  |  |
| --- | --- | --- | --- |
| Section | Question | Mark | Effort (minutes) |
| 1. Sequences | 1 | 6 | 0 |
| 1. Add new service | 1 | 40 | 0 |
| 1. hvk\_bill\_vw | 2 | 10 | 0 |
|  | 5 | 44 | 0 |
| English and handed in properly (coding/naming standards, all output messages are meaningful and clear, effort tracking, self assessment) |  | 7 | 0 |
| Totals |  | 107 | 0 |

**For each of the blocks created for the following problems, include a comment block at the top of the block with the question number, your name and a brief description of what the block does. Insert comments throughout to explain the steps.**

**Use ISO/ANSI standards joins for all joins.**

**Name all identifiers according to the naming standards shown below.**

**Format all blocks using the SQL Developer Formatter.**

**Provide sample output.**

**Naming Standards:**

|  |  |
| --- | --- |
| **Identifier type** | **Prefix** |
| local variables | lv\_ |
| local constant | lc\_ |
| local record | lrec\_ |
| local cursor | lcur\_ |

# Sequences

## Provide the SQL script (i.e. SQL only, no PL/SQL required) to create the following sequences:

| **Sequence Name** | **Starting**  **Value** | **Increment** | **To be used for** |
| --- | --- | --- | --- |
| hvk\_owner\_seq | 250 | 1 | owner\_number |
| hvk\_daily\_rate\_seq | 12 | 1 | daily\_rate\_number |
| hvk\_discount\_seq | 4 | 1 | discount\_number |
| hvk\_food\_seq | 16 | 1 | food\_number |
| hvk\_medication\_seq | 15 | 1 | medication\_number |
| hvk\_pet\_res\_seq | 2500 | 1 | pet\_res\_number |
| hvk\_pet\_seq | 100 | 1 | pet\_number |
| hvk\_reservation\_seq | 2000 | 1 | reservation\_number |
| hvk\_run\_seq | 40 | 1 | run\_number |
| hvk\_service\_seq | 7 | 1 | service\_number |
| hvk\_vaccination\_seq | 7 | 1 | vaccination\_number |
| hvk\_vet\_seq | 10 | 1 | vet\_number |

**SQL:**

**--owner\_number**

**CREATE SEQUENCE hvk\_owner\_seq**

**Start with 250 increment by 1;**

**--daily\_rate\_number**

**CREATE SEQUENCE hvk\_daily\_rate\_seq**

**Start with 12 increment by 1;**

**--discount\_number**

**CREATE SEQUENCE hvk\_discount\_seq**

**Start with 4 increment by 1;**

**--food\_number**

**CREATE SEQUENCE hvk\_food\_seq**

**Start with 16 increment by 1;**

**--medication\_number**

**CREATE SEQUENCE hvk\_medication\_seq**

**Start with 15 increment by 1;**

**--pet\_res\_number**

**CREATE SEQUENCE hvk\_pet\_res\_seq**

**Start with 2500 increment by 1;**

**--pet\_number**

**CREATE SEQUENCE hvk\_pet\_seq**

**Start with 100 increment by 1;**

**--reservation\_number**

**CREATE SEQUENCE hvk\_reservation\_seq**

**Start with 2000 increment by 1;**

**--run\_number**

**CREATE SEQUENCE hvk\_run\_seq**

**Start with 40 increment by 1;**

**--service\_number**

**CREATE SEQUENCE hvk\_service\_seq**

**Start with 7 increment by 1;**

**--vaccination\_number**

**CREATE SEQUENCE hvk\_vaccination\_seq**

**Start with 7 increment by 1;**

**--vet\_number**

**CREATE SEQUENCE hvk\_vet\_seq**

**Start with 10 increment by 1;**

# Add a new service using PL/SQL

## Jim and Sally want to be able to add new services without having to contact you to do it for them. Create an anonymous PL/SQL block to add a new service with the associated rate(s). Use the sequences that you created in part A to generate the primary keys.

### Prompt for the service description. Convert the service to sentence case (first letter uppercase, rest of the letters lowercase.)

### Check to see if the service already exists. If it does print a message stating that the service has already been added and suggest running the change price block to update the prices.

### If the service doesn't exist, prompt for the following:

#### ask if the rate is the same for all dog sizes

#### ask for the basic daily rate (small dog rate for different prices)

#### ask for the percentage increase between dog sizes (must be between 0 and 50%. It is 0 if the price is the same for all dog sizes)

### Add the service to the service table. If the insert fails, display an appropriate error message.

### If the daily rate is the same for all dog sizes, add the basic daily rate for the service. If the insert fails, display an appropriate error message.

### If the daily rate is the different for each dog size, use a loop to add the daily rates. Use a case statement to determine which dog size to add. Increase the daily rate by the given percentage for each dog size. If any of the inserts fail, display an appropriate error message.

### If all the inserts work, display a message with informing the user of the service number and that the daily rate(s) have been added.

### If any of the inserts fail, roll back the entire transaction. Otherwise commit the transaction.

**SQL:**

**--------------------------------------------------------------------------------**

**-----------------------------------PART B---------------------------------------**

**--------------------------------------------------------------------------------**

**SET DEFINE ON;**

**SET SERVEROUTPUT ON;**

**DECLARE**

**lv\_number hvk\_service.service\_number%TYPE;**

**lv\_desc hvk\_service.service\_description%TYPE := '&Service\_Description';**

**lv\_found NUMBER;**

**lv\_rate\_num hvk\_daily\_rate.daily\_rate\_number%TYPE;**

**lv\_same\_rate BOOLEAN;**

**lv\_basic\_rate hvk\_daily\_rate.daily\_rate%TYPE;**

**lv\_increase NUMBER;**

**lv\_size hvk\_daily\_rate.daily\_rate\_dog\_size%TYPE;**

**lv\_ser\_seq number;**

**lv\_dr\_seq number;**

**BEGIN**

**SELECT hvk\_service\_seq.NEXTVAL INTO lv\_ser\_seq FROM DUAL;**

**SELECT hvk\_daily\_rate\_seq.NEXTVAL INTO lv\_dr\_seq FROM DUAL;**

**lv\_desc := UPPER(SUBSTR(lv\_desc, 0, 1)) || LOWER(SUBSTR(lv\_desc, 2));**

**SELECT COUNT(\*) INTO lv\_FOUND**

**FROM hvk\_service**

**WHERE service\_description = lv\_desc;**

**IF lv\_found = 0 THEN**

**lv\_same\_rate := 'N' = UPPER('&Same\_rate\_for\_all\_dog\_sizes?');**

**lv\_basic\_rate := &Basic\_daily\_rate;**

**lv\_increase := &Increase\_between\_dog\_sizes;**

**lv\_increase := (lv\_increase/100)+1;**

**INSERT INTO hvk\_service (**

**service\_number, service\_description**

**) VALUES (**

**hvk\_service\_seq.CURRVAL, lv\_desc**

**);**

**IF lv\_same\_rate THEN**

**INSERT INTO hvk\_daily\_rate (**

**daily\_rate\_number, daily\_rate,**

**daily\_rate\_dog\_size, serv\_service\_number**

**) VALUES (**

**hvk\_daily\_rate\_seq.CURRVAL, lv\_basic\_rate,**

**NULL, hvk\_service\_seq.CURRVAL**

**);**

**ELSE**

**FOR lv\_loop in 0..2 LOOP**

**CASE lv\_loop**

**WHEN 0 THEN**

**INSERT INTO hvk\_daily\_rate (**

**DAILY\_RATE\_NUMBER, DAILY\_RATE,**

**DAILY\_RATE\_DOG\_SIZE, SERV\_SERVICE\_NUMBER**

**) VALUES (**

**hvk\_daily\_rate\_seq.CURRVAL, lv\_basic\_rate,**

**'S', hvk\_service\_seq.CURRVAL**

**);**

**WHEN 1 THEN**

**INSERT INTO hvk\_daily\_rate (**

**daily\_rate\_number, daily\_rate,**

**daily\_rate\_dog\_size, serv\_service\_number**

**) VALUES (**

**hvk\_daily\_rate\_seq.CURRVAL, lv\_basic\_rate \* lv\_increase,**

**'M', hvk\_service\_seq.CURRVAL**

**);**

**WHEN 2 THEN**

**INSERT INTO hvk\_daily\_rate (**

**daily\_rate\_number, daily\_rate,**

**daily\_rate\_dog\_size, serv\_service\_number**

**) VALUES (**

**hvk\_daily\_rate\_seq.CURRVAL, lv\_basic\_rate \* lv\_increase \* lv\_increase,**

**'L', hvk\_service\_seq.CURRVAL**

**);**

**END CASE;**

**END LOOP;**

**END IF;**

**DBMS\_OUTPUT.PUT\_LINE('The service '|| lv\_desc || 'has been added.');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('The service already exists. Maybe you want to update the prices?');**

**END IF;**

**END;**

**--ROLLBACK;**

**Sample output(service doesn’t already exist):**

**Description: new service**

**Same rate: no**

**Daily rate: 3**

**Increase: 0**

The service New service has been added.

**Sample output(service already exists):**

The service already exists. Maybe you want to update the prices?

## 

# Display a bill

## Create a view called **hvk\_bill\_vw** which contains the billing information for contracts. The rows of the view should provide the following information about each service provided during the stay:

* the reservation number
* the owner number
* the service description
* the pet reservation number
* the cost for one day
* the number of days charged for the service
* the total amount charged for the service.

## Test your view on the **hvk\_bill\_vw** test cases in **Appendix I**. Provide your create view command:

**SQL:**

**DROP VIEW hvk\_bill\_vw;**

**CREATE VIEW hvk\_bill\_vw**

**AS**

**SELECT r.reservation\_number AS "RESERVATION\_NUMBER",**

**o.owner\_number AS "OWNER\_NUMBER",**

**s.service\_description AS "SERVICE\_DESC",**

**pr.pet\_res\_number AS "PET\_RES\_NUMBER",**

**dr.daily\_rate AS "DAILY\_RATE",**

**r.reservation\_end\_date - r.reservation\_start\_date AS "DAYS\_CHARGED",**

**(r.reservation\_end\_date - r.reservation\_start\_date) \* dr.daily\_rate**

**AS "AMOUNT\_CHARGED"**

**FROM hvk\_reservation r, hvk\_owner o, hvk\_service s, hvk\_pet\_reservation pr,**

**hvk\_daily\_rate dr, hvk\_pet p, hvk\_pet\_reservation\_service prs**

**WHERE o.owner\_number = p.own\_owner\_number**

**AND p.pet\_number = pr.pet\_pet\_number**

**AND pr.res\_reservation\_number = r.reservation\_number**

**AND pr.pet\_res\_number = prs.pr\_pet\_res\_number**

**AND s.service\_number = prs.serv\_service\_number**

**AND s.service\_number = dr.serv\_service\_number**

**AND p.dog\_size = dr.daily\_rate\_dog\_size**

**AND prs.service\_frequency IS NULL**

**UNION**

**SELECT r.reservation\_number AS "RESERVATION\_NUMBER",**

**o.owner\_number AS "OWNER\_NUMBER",**

**s.service\_description AS "SERVICE\_DESC",**

**pr.pet\_res\_number AS "PET\_RES\_NUMBER",**

**dr.daily\_rate AS "DAILY\_RATE",**

**prs.service\_frequency AS "DAYS\_CHARGED",**

**prs.service\_frequency \* dr.daily\_rate AS "AMOUNT\_CHARGED"**

**FROM hvk\_reservation r, hvk\_owner o, hvk\_service s, hvk\_pet\_reservation pr,**

**hvk\_daily\_rate dr, hvk\_pet p, hvk\_pet\_reservation\_service prs**

**WHERE o.owner\_number = p.own\_owner\_number**

**AND p.pet\_number = pr.pet\_pet\_number**

**AND pr.res\_reservation\_number = r.reservation\_number**

**AND pr.pet\_res\_number = prs.pr\_pet\_res\_number**

**AND s.service\_number = prs.serv\_service\_number**

**AND s.service\_number = dr.serv\_service\_number**

**AND p.dog\_size = dr.daily\_rate\_dog\_size**

**AND prs.service\_frequency IS NOT NULL**

**UNION**

**SELECT r.reservation\_number AS "RESERVATION\_NUMBER",**

**o.owner\_number AS "OWNER\_NUMBER",**

**s.service\_description AS "SERVICE\_DESC",**

**pr.pet\_res\_number AS "PET\_RES\_NUMBER",**

**dr.daily\_rate AS "DAILY\_RATE",**

**r.reservation\_end\_date - r.reservation\_start\_date AS "DAYS\_CHARGED",**

**(r.reservation\_end\_date - r.reservation\_start\_date) \* dr.daily\_rate**

**AS "AMOUNT\_CHARGED"**

**FROM hvk\_reservation r, hvk\_owner o, hvk\_service s, hvk\_pet\_reservation pr,**

**hvk\_daily\_rate dr, hvk\_pet p, hvk\_pet\_reservation\_service prs**

**WHERE o.owner\_number = p.own\_owner\_number**

**AND p.pet\_number = pr.pet\_pet\_number**

**AND pr.res\_reservation\_number = r.reservation\_number**

**AND pr.pet\_res\_number = prs.pr\_pet\_res\_number**

**AND s.service\_number = prs.serv\_service\_number**

**AND s.service\_number = dr.serv\_service\_number**

**AND dr.daily\_rate\_dog\_size IS NULL;**

**Sample output:**

**Run the following select and show your output**

select \* from HVK\_bill\_vw

where RESERVATION\_NUMBER in (102,103,105,106) order by reservation\_number;

## 

## Write an anonymous PL/SQL block that uses the **hvk\_bill\_vw** to display the billing information for a specific reservation. It should display:

### the owner name

### the total before any discounts

### the total amount discounted for each discount type

### the total after discounts

### the GST owing and

### the total amount owing for a reservation.

If the reservation does not exist in **hvk\_bill\_vw** an appropriate error message should be displayed.

The GST rate is 5%. It should be declared as a constant.

All monetary amounts should be displayed in currency format.

The output should look similar to:

Owner: Peter Piper

Total be hvk\_fore discounts: $739.00

Shared Run discount: $40.80

Multiple Pets discount: $51.73

Own Food discount: $40.80

Total after discounts: $605.67

GST: $30.28

Total bill: $635.95

## Test the block using the Billing Information test cases in the **Appendix II**.\

## Provide your program block and sample output:

**SQL:**

**SET SERVEROUTPUT ON;**

**DECLARE**

**lv\_res\_num hvk\_reservation.reservation\_number%TYPE := &reservation\_number;**

**lv\_total\_before hvk\_daily\_rate.daily\_rate%TYPE;**

**lv\_share\_run hvk\_daily\_rate.daily\_rate%TYPE;**

**lv\_3\_more\_pets hvk\_daily\_rate.daily\_rate%TYPE;**

**lv\_own\_food hvk\_daily\_rate.daily\_rate%TYPE;**

**lv\_total\_after hvk\_daily\_rate.daily\_rate%TYPE;**

**lv\_gst hvk\_daily\_rate.daily\_rate%TYPE;**

**lv\_total hvk\_daily\_rate.daily\_rate%TYPE;**

**lv\_own\_name VARCHAR2(50);**

**lv\_gst\_rate CONSTANT hvk\_daily\_rate.daily\_rate%TYPE := 0.05;**

**BEGIN**

**SELECT DISTINCT o.owner\_first\_name || ' ' ||o.owner\_last\_name INTO lv\_own\_name**

**FROM hvk\_owner o, hvk\_bill\_vw b**

**WHERE b.reservation\_number = lv\_res\_num AND b.owner\_number = o.owner\_number;**

**SELECT NVL((SUM(b.amount\_charged)),0) INTO lv\_total\_before**

**FROM hvk\_bill\_vw b**

**WHERE b.reservation\_number = lv\_res\_num;**

**SELECT NVL((SUM(b.amount\_charged) \* 0.10),0) into lv\_share\_run**

**FROM hvk\_bill\_vw b, hvk\_pet\_reservation pr1**

**WHERE b.reservation\_number = lv\_res\_num**

**AND b.service\_desc = 'Boarding'**

**AND pr1.pet\_res\_number = b.pet\_res\_number**

**AND pr1.pr\_sharing\_with IS NOT NULL;**

**SELECT NVL((SUM(b.amount\_charged) \* 0.07) ,0) into lv\_3\_more\_pets**

**FROM hvk\_bill\_vw b**

**WHERE b.reservation\_number = lv\_res\_num**

**AND ( SELECT count(pr.res\_reservation\_number)**

**FROM hvk\_pet\_reservation pr**

**WHERE pr.res\_reservation\_number = lv\_res\_num ) >= 3;**

**SELECT NVL((SUM(b.amount\_charged) \* 0.10 ), 0.00) INTO lv\_own\_food**

**FROM hvk\_bill\_vw b, hvk\_pet\_food pf, hvk\_pet p, hvk\_pet\_reservation pr**

**WHERE b.reservation\_number = lv\_res\_num**

**AND pr.pet\_res\_number = b.pet\_res\_number**

**AND pr.pet\_pet\_number = p.pet\_number**

**AND p.dog\_size = 'L'**

**AND b.pet\_res\_number = pf.pr\_pet\_res\_number**

**AND pf.food\_food\_number = 13**

**AND b.service\_desc = 'Boarding';**

**lv\_total\_after := lv\_total\_before - lv\_share\_run - lv\_3\_more\_pets - lv\_own\_food;**

**lv\_gst := lv\_total\_after \* lv\_gst\_rate;**

**lv\_total := lv\_total\_after + lv\_gst;**

**DBMS\_OUTPUT.PUT\_LINE('Owner: ' || lv\_own\_name);**

**DBMS\_OUTPUT.PUT\_LINE('Total before discounts: ' || to\_char(lv\_total\_before, '$9,999.99'));**

**DBMS\_OUTPUT.PUT\_LINE('Shared run discount: ' || to\_char(lv\_share\_run, '$9,999.99'));**

**DBMS\_OUTPUT.PUT\_LINE('Multiple pets discount: ' || to\_char(lv\_3\_more\_pets, '$9,999.99'));**

**DBMS\_OUTPUT.PUT\_LINE('Own food discount: ' || to\_char(lv\_own\_food, '$9,999.99'));**

**DBMS\_OUTPUT.PUT\_LINE('Total after discounts: ' || to\_char(lv\_total\_after, '$9,999.99'));**

**DBMS\_OUTPUT.PUT\_LINE('-------------------------------------------------');**

**DBMS\_OUTPUT.PUT\_LINE('GST: ' || to\_char(lv\_gst, '$9,999.99'));**

**DBMS\_OUTPUT.PUT\_LINE('-------------------------------------------------');**

**DBMS\_OUTPUT.PUT\_LINE('Total bill: ' || to\_char(lv\_total, '$9,999.99'));**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**IF lv\_own\_name IS NULL THEN**

**DBMS\_OUTPUT.PUT\_LINE('ERROR: There is no billing information for reservation ' || lv\_res\_num);**

**END IF;**

**END;**

**Output:**

**Show sample output for res num 110:**

**Owner: Peter Piper**

**Total before discounts: $739.00**

**Shared run discount: $40.80**

**Multiple pets discount: $51.73**

**Own food discount: $40.80**

**Total after discounts: $605.67**

**-------------------------------------------------**

**GST: $30.28**

**-------------------------------------------------**

**Total bill: $635.95**

**Fill in the following table (Test Run, Test Passed)**

| **TC**  **#** | **res**  **num** | **Test Run (✓ )** | **Test Passed (✓ )** | **Expected Results (grand total)** |
| --- | --- | --- | --- | --- |
| 1 | 105 | **✓** | **✓** | $219.45 |
| 2 | 109 | **✓** | **✓** | $214.20 |
| 3 | 114 | **✓** | **✓** | $39.90 |
| 4 | 106 | **✓** | **✓** | $184.70 |
| 5 | 802 | **✓** | **✓** | $156.24 |
| 6 | 800 | **✓** | **✓** | $122.85 |
| 7 | 110 | **✓** | **✓** | $635.95 |
| 8 | 999 | **✓** | **✓** | ERROR: There is no billing information for reservation 999. |