

ANIMAL SHELTER DATABASE SYSTEM PROJECT - PART 5

Updated Assumptions:

1. Animals can be transferred from one branch to another
2. Each shelter branch will have multiple employees
3. Each shelter branch will have a branch manager
4. Employees work only in one shelter branch only (no transfer of employees)
5. An animal can be adopted/surrendered to the shelter for various reasons multiple times
6. A vaccination is given each time an animal is detected with any disease
7. Sponsors can support multiple shelter branches
8. Full payment must be done in a single payment for adoption of an animal
9. Each medical condition of an animal is treated by only one veterinarian
10. Veterinarians can treat multiple conditions of multiple animals
11. Each transfer of an animal involves transfer from one branch to another
12. Each veterinary doctor can have multiple specializations
13. All contacts should not contain spaces
14. Employees work on hourly basis and hourly pay varies from employee to employee
15. Any animal is considered as ready for adoption if its vaccination status is up to date
16. Gross pay is the total pay for an employee without any deduction
17. Net pay is the pay of an employee after tax deduction
18. Employee salary is an estimate considering the maximum number of hours they can work
19. Discount is applied on adoption fee if the amount is greater than 750 by 5% and when 1000 by 10%
20. Surrender reason cannot exceed 20 characters
21. Date of birth of an animal cannot be greater than date of acquisition

Screenshots of PL/SQL code

Procedures

1] Creating a procedure to update the food quantity for animals with up to date vaccination status

The screenshot shows the Oracle Database Actions interface. The left sidebar displays a tree view of database objects, including tables like ACCOMMODATE, ADOPTION_DETAILS, ANIMAL_FEATURES, ANIMAL_INFO, ANIMAL_TRANSFER_RECORDS, CONTAIN, DONE_BY, EMPLOYEES_STAFF_DETAILS, EMPLOYEE_PAYROLL, EXAMINE, FACILITIES, GROOMING_HISTORY, MEDICAL_HISTORY, NEED, NUTRITION, PAYMENT_DETAILS, and PERFORM. The main editor displays the following PL/SQL code:

```
1 CREATE OR REPLACE PROCEDURE update_food_quantity(changed_quantity INT) AS
2 BEGIN
3   FOR rec IN (
4     SELECT a.animal_id, n.food_id, n.food_quantity, a.a_vaccine_status
5     FROM animal_info a
6     JOIN need ON a.animal_id = need.animal_id
7     JOIN nutrition n ON need.food_id = n.food_id
8   )
9   LOOP
10    IF rec.a_vaccine_status = 'Up to date' THEN
11      UPDATE nutrition
12      SET food_quantity = food_quantity - changed_quantity
13      WHERE food_id = rec.food_id;
14      DBMS_OUTPUT.PUT_LINE('Food quantity updated for animal ' || rec.animal_id || ' : ' || rec.food_quantity - changed_quantity);
15    END IF;
16  END LOOP;
17 END;
```

The bottom panel shows the execution results:

- Query Result: 1 row inserted.
- Elapsed: 00:00:00.007
- Procedure UPDATE_FOOD_QUANTITY compiled
- Elapsed: 00:00:00.101

The screenshot shows the Oracle Database Actions interface. The left sidebar displays a tree view of database objects, including tables like ACCOMMODATE, ADOPTION_DETAILS, ANIMAL_FEATURES, ANIMAL_INFO, ANIMAL_TRANSFER_RECORDS, CONTAIN, DONE_BY, EMPLOYEES_STAFF_DETAILS, EMPLOYEE_PAYROLL, EXAMINE, FACILITIES, GROOMING_HISTORY, MEDICAL_HISTORY, NEED, NUTRITION, PAYMENT_DETAILS, and PERFORM. The main editor displays the following PL/SQL code:

```
1 BEGIN
2   update_food_quantity(2);
3 END;
```

The bottom panel shows the execution results:

- Query Result: Food quantity updated for animal 1: 96, Food quantity updated for animal 4: 76, Food quantity updated for animal 7: 36, Food quantity updated for animal 10: 61.
- PL/SQL procedure successfully completed.
- Elapsed: 00:00:00.023

2] Creating a procedure to get the details of Veterinarians with more than x years of work experience

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database schema with various tables like ACCOMMODATE, ADOPTION_DETAILS, ANIMAL_FEATURES, etc. The main window shows a SQL script titled 'GetVeterinariansByExperience'.

```
1 CREATE OR REPLACE PROCEDURE GetVeterinariansByExperience(  
2   p_min_experience IN INT,  
3   p_cursor OUT SYS_REFCURSOR  
4 ) AS  
5 BEGIN  
6   OPEN p_cursor FOR  
7   SELECT *  
8   FROM veterinary  
9   WHERE work_exp >= p_min_experience;  
10 END  
11 GetVeterinariansByExperience;  
12 /
```

The bottom pane shows the 'Script Output' tab with the following message:

```
PL/SQL procedure successfully completed.  
Elapsed: 00:00:00.023  
  
Procedure GETVETERINARIANSBYEXPERIENCE compiled  
Elapsed: 00:00:00.035
```

The status bar at the bottom indicates the session is executed by ADMIN at 2:06:31 AM.

The screenshot shows the same Oracle SQL Developer interface, but now the procedure is being executed. The SQL script in the main window is as follows:

```
1 DECLARE  
2   vet_cursor SYS_REFCURSOR;  
3   v_vet_id veterinary.vet_id%TYPE;  
4   v_vet_name veterinary.vet_name%TYPE;  
5   v_vet_contact veterinary.vet_contact%TYPE;  
6   v_education veterinary.education%TYPE;  
7   v_work_exp veterinary.work_exp%TYPE;  
8 BEGIN  
9   GetVeterinariansByExperience(10, vet_cursor);  
10  
11 LOOP  
12   FETCH vet_cursor INTO v_vet_id, v_vet_name, v_vet_contact, v_education, v_work_exp;  
13   EXIT WHEN vet_cursor%NOTFOUND;  
14   DBMS_OUTPUT.PUT_LINE('Veterinarian ID: ' || v_vet_id || ', Name: ' || v_vet_name || ', Contact: ' || v_vet_contact || ',  
15   Education: ' || v_education || ', Experience: ' || v_work_exp);  
16 END LOOP;  
17 CLOSE vet_cursor;  
18 END;  
19 /
```

The 'Script Output' tab shows the following output:

```
PL/SQL procedure successfully completed.  
Elapsed: 00:00:00.018  
  
Veterinarian ID: 9, Name: Dr. Anderson, Contact: +1777777777,  
Education: Small Animal Internal Medicine, Experience: 20
```

The status bar at the bottom indicates the session is executed by ADMIN at 2:07:47 AM.

The screenshot shows the Oracle Database Actions interface. The left sidebar contains a 'Navigator' pane with a tree view of database objects. The main area displays the execution of a PL/SQL procedure named 'GetVeterinariansByExperience'. The procedure code is shown in the top editor, and the results are displayed in the 'Query Result' pane below. The results list 10 veterinarians with their IDs, names, contact information, education, and experience.

```

1 DECLARE
2   vet_cursor SYS_REFCURSOR;
3   v_vet_id veterinary.vet_id%TYPE;
4   v_vet_name veterinary.vet_name%TYPE;
5   v_vet_contact veterinary.vet_contact%TYPE;
6   v_education veterinary.education%TYPE;
7   v_work_exp veterinary.work_exp%TYPE;
8 BEGIN
9   GetVeterinariansByExperience(10, vet_cursor);
10

```

Query Result

Veterinarian ID	Name	Contact	Education	Experience
1	Dr. Smith	+1234567890	Doctor of Veterinary Medicine	10
3	Dr. Williams	+1555555555	Veterinary Surgery	12
4	Dr. Brown	+1666666666	Veterinary Pathobiology	15
5	Dr. Wilson	+1777777777	Veterinary Oncology	18
7	Dr. Martinez	+1999999999	Veterinary Dermatology	11
8	Dr. Garcia	+1888888888	Veterinary Radiology	14
9	Dr. Anderson	+1777777777	Small Animal Internal Medicine	20

PL/SQL procedure successfully completed.
Elapsed: 00:00:00.018

3) Creating a procedure to display the vaccinations given on a particular date

The screenshot shows the Oracle Database Actions interface. The left sidebar contains a 'Navigator' pane with a tree view of database objects. The main area displays the execution of a PL/SQL procedure named 'DisplayVaccinationsForDate'. The procedure code is shown in the top editor, and the results are displayed in the 'Query Result' pane below. The results show the execution of the procedure and the time taken to compile it.

```

1 CREATE OR REPLACE PROCEDURE DisplayVaccinationsForDate(
2   p_vaccination_date IN DATE
3 ) AS
4 BEGIN
5   FOR rec IN (
6     SELECT vaccination_name, vaccination_date, vaccination_dosage
7     FROM vaccination
8     WHERE vaccination_date = p_vaccination_date
9   ) LOOP
10    DBMS_OUTPUT.PUT_LINE('Vaccination Name: ' || rec.vaccination_name);
11    DBMS_OUTPUT.PUT_LINE('Vaccination Date: ' || TO_CHAR(rec.vaccination_date, 'YYYY-MM-DD'));
12    DBMS_OUTPUT.PUT_LINE('Vaccination Dosage: ' || rec.vaccination_dosage);
13    DBMS_OUTPUT.PUT_LINE('-----');
14  END LOOP;
15 END DisplayVaccinationsForDate;
16
17

```

Query Result

Elapsed: 00:00:00.018

Procedure DISPLAYVACCINATIONSFORDATE compiled

Elapsed: 00:00:00.057

The screenshot shows the Oracle Database Actions SQL interface. The left sidebar contains a Navigator with a tree view of database objects, including ADMIN, Tables, and various tables like ACCOMMODATE, ADOPTION_DETAILS, etc. The main editor displays a PL/SQL procedure named `DisplayVaccinationsForDate` with the following code:

```

1 DECLARE
2   v_date DATE := DATE '2023-05-12';
3 BEGIN
4   DisplayVaccinationsForDate(v_date);
5 END;
6 /
7

```

The bottom panel shows the execution results under the "Script Output" tab:

```

Vaccination Name: Feline Panleukopenia
Vaccination Date: 2023-05-12
Vaccination Dosage: Initial
-----
PL/SQL procedure successfully completed.
Elapsed: 00:00:00.009

```

The status bar at the bottom indicates the execution was completed at 2:09:51 AM on 3/27/2024 by user ADMIN.

4) Creating a procedure to get the vaccination name for a particular disease as per historical records.

The screenshot shows the Oracle Database Actions SQL interface. The left sidebar is the same as the previous screenshot. The main editor displays a PL/SQL procedure named `GetVaccinationForDisease` with the following code:

```

1 CREATE OR REPLACE PROCEDURE GetVaccinationForDisease(
2   p_disease IN VARCHAR2,
3   p_vaccination_name OUT VARCHAR2
4 ) AS
5 BEGIN
6   SELECT v.vaccination_name
7   INTO p_vaccination_name
8   FROM vaccination v
9   JOIN medical_history m ON v.vaccination_id = m.vaccination_id
10  WHERE m.disease = p_disease;
11 EXCEPTION
12  WHEN NO_DATA_FOUND THEN
13    DBMS_OUTPUT.PUT_LINE('No vaccination found for the provided disease.');

The bottom panel shows the execution results under the "Script Output" tab:



```

PL/SQL procedure successfully completed.
Elapsed: 00:00:00.009

Procedure GETVACCINATIONFORDISEASE compiled
Elapsed: 00:00:00.026

```



The status bar at the bottom indicates the execution was completed at 2:10:44 AM on 3/27/2024 by user ADMIN.


```

The screenshot shows the Oracle Database Actions interface. The left sidebar contains a 'Navigator' pane with a tree view of database objects. The main area displays the SQL editor with the following code:

```

1 DECLARE
2   v_vaccination_name vaccination.vaccination_name%TYPE;
3   v_disease VARCHAR2(25) := 'Infection';
4 BEGIN
5   GetVaccinationForDisease(v_disease, v_vaccination_name);
6
7   DBMS_OUTPUT.PUT_LINE('Vaccination Name for ' || v_disease || ' : ' || v_vaccination_name);
8
9
10

```

Below the editor, the 'Script Output' tab shows the execution results:

```

Elapsed: 00:00:00.018

Vaccination Name for Infection: Feline Calicivirus

PL/SQL procedure successfully completed.
Elapsed: 00:00:00.012

```

The bottom status bar indicates the execution was completed at 2:12:47 AM by ADMIN.

5] Creating a procedure to get the list of all the shelters with capacity more than the given number

The screenshot shows the Oracle Database Actions interface. The left sidebar contains a 'Navigator' pane with a tree view of database objects. The main area displays the SQL editor with the following code:

```

1 CREATE OR REPLACE PROCEDURE GetSheltersWithCapacity(
2   p_capacity_threshold IN INT
3 ) AS
4 BEGIN
5   FOR shelter_rec IN (
6     SELECT sb.shelter_name
7     FROM shelter_branch sb
8     JOIN facilities f ON sb.shelter_id = f.shelter_id
9     WHERE f.capacity > p_capacity_threshold
10  ) LOOP
11     DBMS_OUTPUT.PUT_LINE('Shelter Name: ' || shelter_rec.shelter_name);
12  END LOOP;
13 END GetSheltersWithCapacity;
14
15
16
17

```

Below the editor, the 'Script Output' tab shows the execution results:

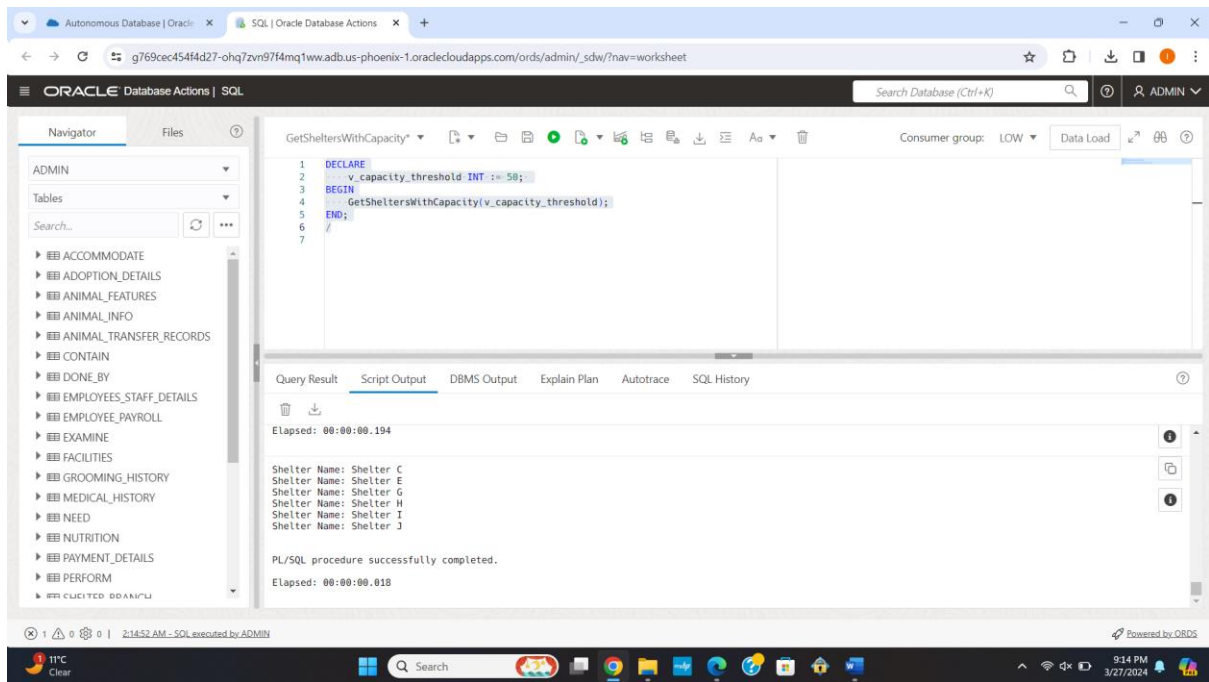
```

PL/SQL procedure successfully completed.
Elapsed: 00:00:00.012

Procedure GETSHELTERSWITHCAPACITY compiled
Elapsed: 00:00:00.194

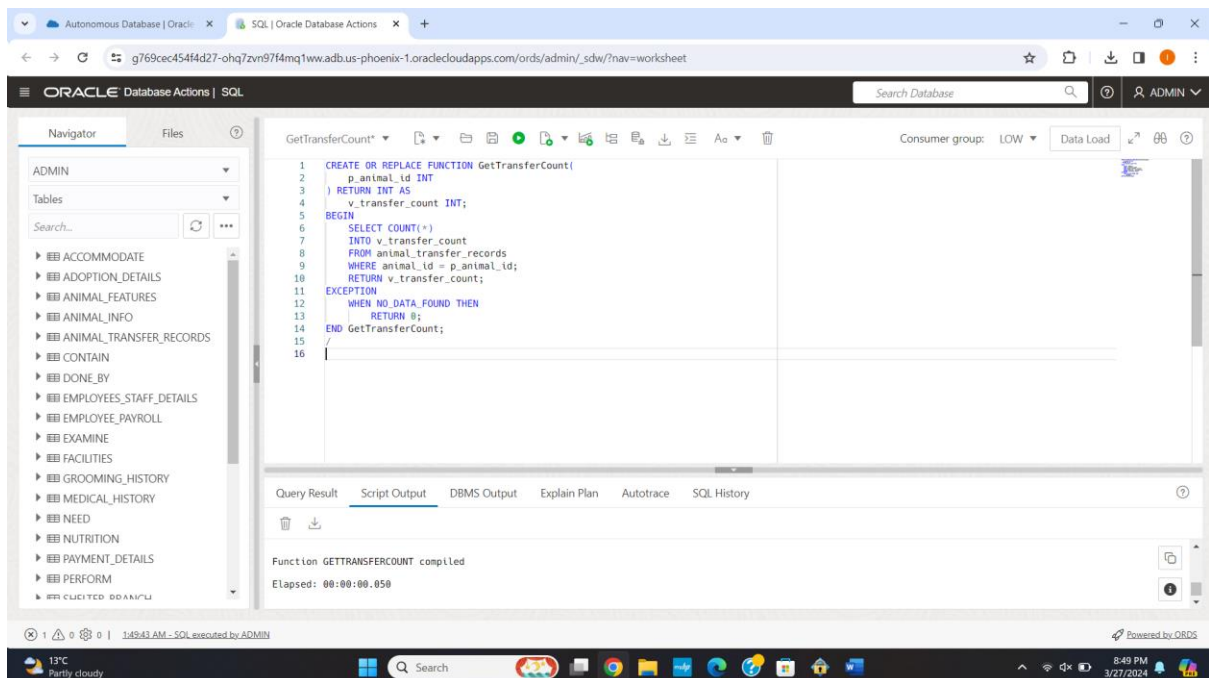
```

The bottom status bar indicates the execution was completed at 2:13:55 AM by ADMIN.



Functions:

1] Creating a function to get the count of the number of times an animal is transferred from one shelter to another.



Autonomous Database | Oracle | SQL | Oracle Database Actions

g769cec454f4d27-ohq7zn97f4mq1ww.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/_sdw/?nav=worksheet

ORACLE Database Actions | SQL

Search Database (Ctrl+K)

ADMIN

Tables

Search...

► ACCOMMODATE
► ADOPTION_DETAILS
► ANIMAL_FEATURES
► ANIMAL_INFO
► ANIMAL_TRANSFER_RECORDS
► CONTAIN
► DONE_BY
► EMPLOYEES_STAFF_DETAILS
► EMPLOYEE_PAYROLL
► EXAMINE
► FACILITIES
► GROOMING_HISTORY
► MEDICAL_HISTORY
► NEED
► NUTRITION
► PAYMENT_DETAILS
► PERFORM
► QUICKEST_DOGANALYSIS

GetTransferCount*
1 SELECT GetTransferCount(3) AS transfer_count FROM dual;

Consumer group: LOW Data Load

Query Result Script Output DBMS Output Explain Plan Autotrace SQL History

Download Execution time: 0.022 seconds

TRANSFER_COUNT
1

1 1

13°C Partly cloudy 8:50 PM 3/27/2024

2] Creating a function to get the salary of a given employee

Autonomous Database | Oracle | SQL | Oracle Database Actions

g769cec454f4d27-ohq7zn97f4mq1ww.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/_sdw/?nav=worksheet

ORACLE Database Actions | SQL

Search Database

ADMIN

Tables

Search...

► ACCOMMODATE
► ADOPTION_DETAILS
► ANIMAL_FEATURES
► ANIMAL_INFO
► ANIMAL_TRANSFER_RECORDS
► CONTAIN
► DONE_BY
► EMPLOYEES_STAFF_DETAILS
► EMPLOYEE_PAYROLL
► EXAMINE
► FACILITIES
► GROOMING_HISTORY
► MEDICAL_HISTORY
► NEED
► NUTRITION
► PAYMENT_DETAILS
► PERFORM
► QUICKEST_DOGANALYSIS

GetEmployeeSalary*
1 CREATE OR REPLACE FUNCTION GetEmployeeSalary(
2 p_emp_id INT
3) RETURN NUMERIC AS
4 v_salary NUMERIC;
5 BEGIN
6 SELECT emp_salary
7 INTO v_salary
8 FROM employees_staff_details
9 WHERE emp_id = p_emp_id;
10 RETURN v_salary;
11 EXCEPTION
12 WHEN NO_DATA_FOUND THEN
13 RETURN NULL;
14 END GetEmployeeSalary;
15 /
16

Consumer group: LOW Data Load

Query Result Script Output DBMS Output Explain Plan Autotrace SQL History

Function GETEMPLOYEESALARY compiled
Elapsed: 00:00:00.069

1 15:05 AM - SQL executed by ADMIN

13°C Partly cloudy 8:51 PM 3/27/2024

The screenshot shows the Oracle Database Actions SQL interface. The left sidebar contains a 'Navigator' with a tree view of database objects under the 'ADMIN' schema, including tables like ACCOMMODATE, ADOPTION_DETAILS, ANIMAL_FEATURES, ANIMAL_INFO, ANIMAL_TRANSFER_RECORDS, CONTAIN, DONE_BY, EMPLOYEES_STAFF_DETAILS, EMPLOYEE_PAYROLL, EXAMINE, FACILITIES, GROOMING_HISTORY, MEDICAL_HISTORY, NEED, NUTRITION, PAYMENT_DETAILS, and PERFORM. The main editor displays a SQL query: `SELECT GetEmployeeSalary(3) AS salary FROM dual;`. Below the editor, the 'Query Result' tab is active, showing a single row with the column 'SALARY' and the value '55000'. The execution time is 0.038 seconds. The bottom status bar shows the time as 1:51:34 AM and the user as ADMIN.

3] Creating a function to check where outdoor space is available for a given shelter ID

The screenshot shows the Oracle Database Actions SQL interface. The left sidebar is the same as in the previous screenshot. The main editor displays a SQL script to create a function: `CREATE OR REPLACE FUNCTION IsOutdoorAvailable(p_shelter_id IN INT) RETURN VARCHAR2 AS
2 outdoor_status VARCHAR2(3);
3 BEGIN
4 SELECT CASE WHEN outdoor_space = 'Y' THEN 'Yes' ELSE 'No' END
5 INTO outdoor_status
6 FROM facilities
7 WHERE shelter_id = p_shelter_id;
8 RETURN outdoor_status;
9 END;`. Below the editor, the 'Script Output' tab is active, showing the execution details: 'Elapsed: 00:00:00.069' and 'Function ISOUTDOORAVAILABLE compiled' with an elapsed time of '00:00:00.073'. The bottom status bar shows the time as 1:52:17 AM and the user as ADMIN.

The screenshot shows the Oracle Database Actions SQL interface. The left sidebar contains a Navigator with a tree view of database objects, including ADMIN, Tables, and various tables like ACCOMMODATE, ADOPTION_DETAILS, etc. The main editor displays a PL/SQL procedure named `IsOutdoorAvailable`. The procedure declares `v_outdoor_status` as `VARCHAR2(3)` and `v_shelter_id` as `INT`. It begins by setting `v_outdoor_status := IsOutdoorAvailable(v_shelter_id);`. Then, it uses an `IF` statement: `IF v_outdoor_status = 'Yes' THEN DBMS_OUTPUT.PUT_LINE('Outdoor space is available.');` and `ELSE DBMS_OUTPUT.PUT_LINE('Outdoor space is not available.');`. The procedure ends with `END IF;` and `END;`. Below the editor, the 'Script Output' tab shows the results: 'Outdoor space is not available.' and 'PL/SQL procedure successfully completed.' with an elapsed time of 00:00:00.015. The bottom status bar indicates the session is executed by ADMIN at 1:53:25 AM on 3/27/2024.

4) Creating a function to check whether a new animal can be inserted into a shelter considering its capacity according to the facilities.

The screenshot shows the Oracle Database Actions SQL interface. The left sidebar contains a Navigator with a tree view of database objects, including ADMIN, Tables, and various tables like ACCOMMODATE, ADOPTION_DETAILS, etc. The main editor displays a PL/SQL function named `IsSpaceAvailable`. The function is defined as `CREATE OR REPLACE FUNCTION IsSpaceAvailable(p_shelter_id IN INT) RETURN VARCHAR2 AS`. It declares `v_capacity` as `INT` and `v_anl_count` as `INT`. The function begins by selecting `f.capacity` into `v_capacity` from `facilities` where `shelter_id = p_shelter_id`. Then, it selects `COUNT(DISTINCT animal_id)` into `v_anl_count` from `accommodate` where `shelter_id = p_shelter_id`. It then uses an `IF` statement: `IF v_anl_count < v_capacity THEN RETURN 'Space Available';` and `ELSE RETURN 'No Space Available';`. The function ends with `END IF;` and `EXCEPTION WHEN NO_DATA_FOUND THEN RETURN 'Shelter Not Found';`. Below the editor, the 'Script Output' tab shows the results: 'Function ISSPACEAVAILABLE compiled' and 'Elapsed: 00:00:00.194'. The bottom status bar indicates the session is executed by ADMIN at 1:54:20 AM on 3/27/2024.

The screenshot shows the Oracle Database Actions interface. The left sidebar contains a 'Navigator' pane with a tree view of database objects, including 'ADMIN', 'Tables', and various tables like 'ACCOMMODATE', 'ADOPTION_DETAILS', etc. The main editor area is titled 'IsSpaceAvailable*' and contains the following PL/SQL code:

```
1 DECLARE
2   v_shelter_id INT := 2;
3   v_space_status VARCHAR2(20);
4 BEGIN
5   v_space_status := IsSpaceAvailable(v_shelter_id);
6   DBMS_OUTPUT.PUT_LINE('Space Status for Shelter: ' || v_shelter_id || ' : ' || v_space_status);
7 END;
```

The 'Script Output' pane at the bottom shows the results of the execution:

```
Space Status for Shelter 2: Space Available
PL/SQL procedure successfully completed.
Elapsed: 00:00:00.015
```

The status bar at the bottom indicates '15:17 AM - SQL executed by ADMIN' and 'Powered by: ODS'.

5] Creating a function to get the total funds from the sponsors for a particular shelter ID.

The screenshot shows the Oracle Database Actions interface. The left sidebar contains a 'Navigator' pane with a tree view of database objects, including 'ADMIN', 'Tables', and various tables like 'ACCOMMODATE', 'ADOPTION_DETAILS', etc. The main editor area is titled 'GetFundsForShelter*' and contains the following PL/SQL code:

```
1 CREATE OR REPLACE FUNCTION GetFundsForShelter(
2   p_shelter_id IN INT
3 ) RETURN NUMERIC AS
4   total_funds NUMERIC := 0;
5 BEGIN
6   SELECT SUM(amount) INTO total_funds
7   FROM payment_details
8   WHERE sponsor_id IN (
9     SELECT sponsor_id
10    FROM support
11    WHERE shelter_id = p_shelter_id
12  );
13 RETURN total_funds;
14 END;
```

The 'Script Output' pane at the bottom shows the results of the execution:

```
Elapsed: 00:00:00.041
Function GETFUNDSFORSHELTER compiled
Elapsed: 00:00:00.041
```

The status bar at the bottom indicates '15:04 AM - SQL executed by ADMIN' and 'Powered by: ODS'.

The screenshot shows the Oracle Database Actions SQL editor. The left sidebar contains a Navigator with a tree view of database objects under the 'ADMIN' schema, including tables like ACCOMMODATE, ADOPTION_DETAILS, ANIMAL_FEATURES, ANIMAL_INFO, ANIMAL_TRANSFER_RECORDS, CONTAIN, DONE_BY, EMPLOYEES_STAFF_DETAILS, EMPLOYEE_PAYROLL, EXAMINE, FACILITIES, GROOMING_HISTORY, MEDICAL_HISTORY, NEED, NUTRITION, PAYMENT_DETAILS, and PERFORM. The main editor area shows a PL/SQL procedure named 'GetFundsForShelter' with the following code:

```

1 DECLARE
2   v_shelter_id INT := 1;
3   v_total_funds NUMERIC;
4 BEGIN
5   v_total_funds := GetFundsForShelter(v_shelter_id);
6   DBMS_OUTPUT.PUT_LINE('Total funds for shelter ' || v_shelter_id || ' : ' || v_total_funds);
7 END;
8 /
9

```

The bottom panel shows the 'Script Output' tab with the following results:

```

Total funds for shelter 1: 500

PL/SQL procedure successfully completed.
Elapsed: 00:00:00.014

```

The status bar at the bottom indicates the execution was completed at 1:58:54 AM on 3/27/2024 by user ADMIN.

Triggers

1) Creating a trigger to calculate the adoption fee post discount (10% if 1000 or above, 5% if 750 or above) and update the value in table accordingly.

The screenshot shows the Oracle Database Actions SQL editor. The left sidebar contains the same Navigator as the previous screenshot. The main editor area shows a trigger named 'CreateDiscount' with the following code:

```

1 CREATE OR REPLACE TRIGGER CalculateDiscount
2 BEFORE INSERT ON adoption_details
3 FOR EACH ROW
4 DECLARE
5   v_payment_amount adoption_details.ad_fee%TYPE;
6 BEGIN
7   v_payment_amount := :NEW.ad_fee;
8   IF v_payment_amount >= 1000 THEN
9     :NEW.ad_fee := v_payment_amount * 0.9;
10    DBMS_OUTPUT.PUT_LINE('Discount applied: New ad fee is ' || TO_CHAR(:NEW.ad_fee, '99999.99'));
11  ELSIF v_payment_amount >= 750 THEN
12    :NEW.ad_fee := v_payment_amount * 0.95;
13    DBMS_OUTPUT.PUT_LINE('Discount applied: New ad fee is ' || TO_CHAR(:NEW.ad_fee, '99999.99'));
14  END IF;
15 END;
16 /

```

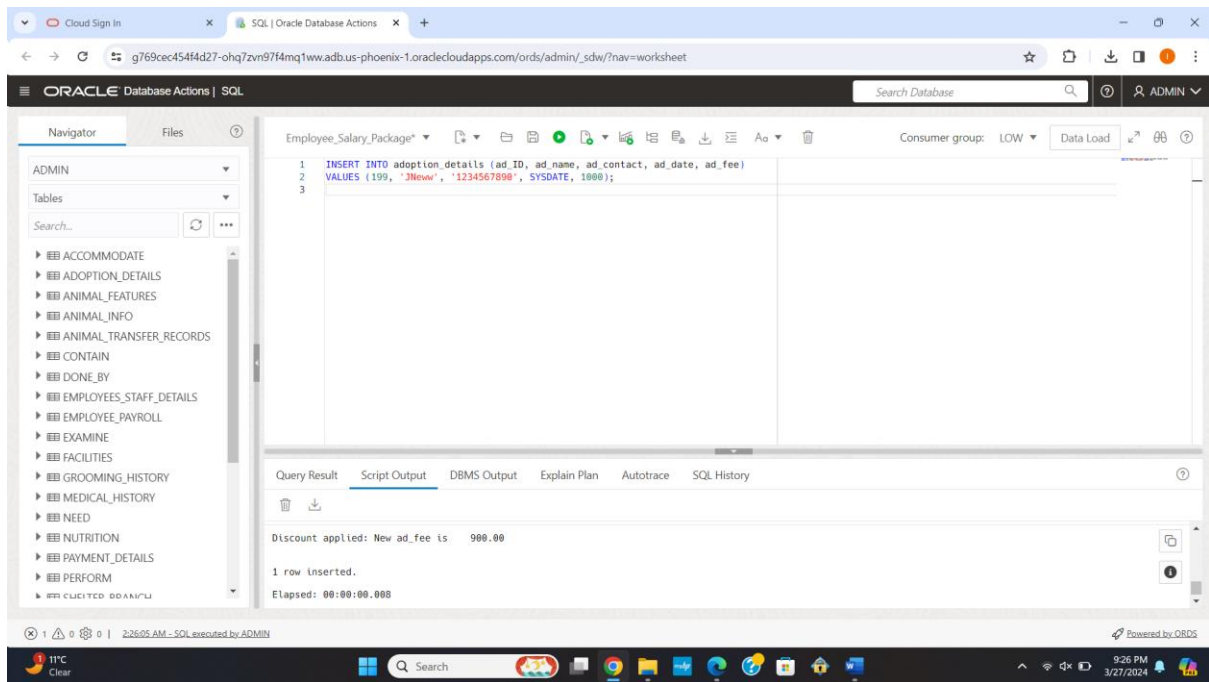
The bottom panel shows the 'Script Output' tab with the following results:

```

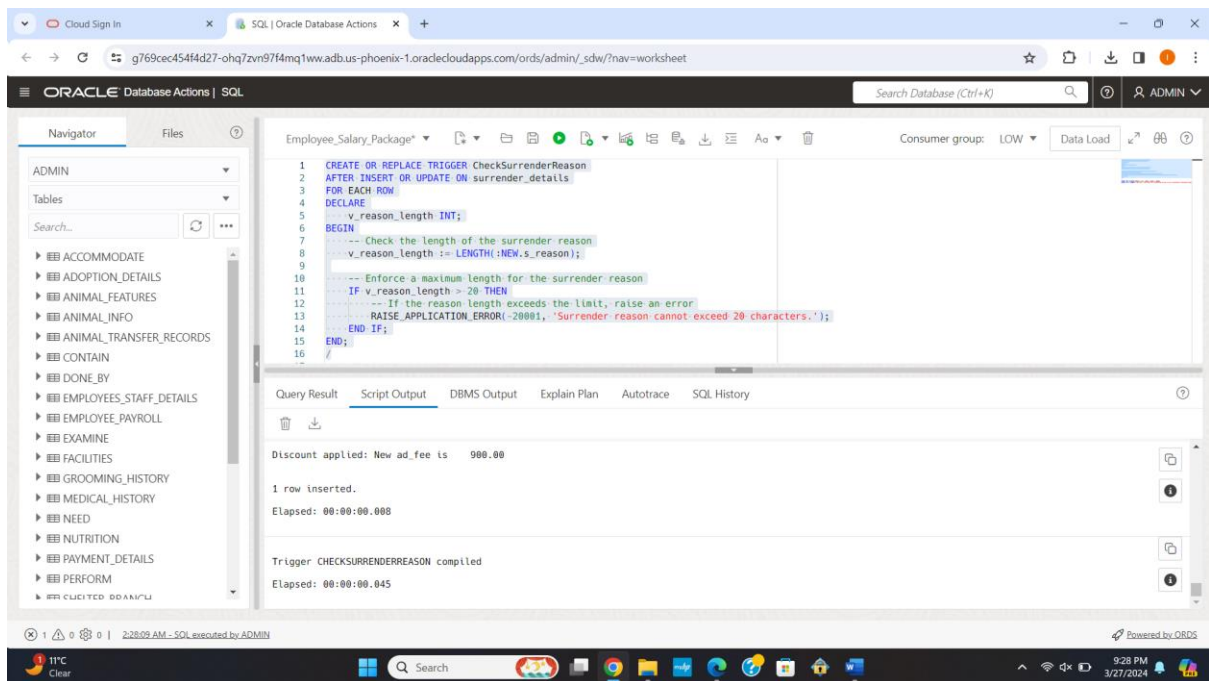
Trigger CALCULATEDISCOUNT compiled
Elapsed: 00:00:00.063

```

The status bar at the bottom indicates the execution was completed at 2:23:28 AM on 3/27/2024 by user ADMIN.



2] Creating a trigger to ensure that the reason for surrendering an animal does not exceed more than 20 characters.



Cloud Sign In | SQL | Oracle Database Actions | +

g769cec454f4d27-ohq7zn97f4mq1ww.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/_sdw/?nav=worksheet

ORACLE Database Actions | SQL

Search Database

ADMIN

Navigator

Files

ADMIN

Tables

Search...

ACCOMMODATE

ADOPTION_DETAILS

ANIMAL_FEATURES

ANIMAL_INFO

ANIMAL_TRANSFER_RECORDS

CONTAIN

DONE_BY

EMPLOYEES_STAFF_DETAILS

EMPLOYEE_PAYROLL

EXAMINE

FACILITIES

GROOMING_HISTORY

MEDICAL_HISTORY

NEED

NUTRITION

PAYMENT_DETAILS

PERFORM

EMPLOYEE_PAYROLL

Employee_Salary_Package*

Consumer group: LOW

Data Load

```
1
2 INSERT INTO surrender_details (s_id, s_date, s_name, s_reason)
3 VALUES (1011, SYSDATE, 'John Doe', 'The reason for surrendering the pet is due to unforeseen circumstances beyond my control,
4 including financial constraints, health issues, and changes in living arrangements');
5
6
```

Query Result

Script Output

DBMS Output

Explain Plan

Autotrace

SQL History

Trigger CHECKSURRENDERREASON compiled

Elapsed: 00:00:00.045

ORA-20001: Surrender reason cannot exceed 20 characters.

ORA-06512: at "ADMIN.CHECKSURRENDERREASON", line 10

ORA-04088: error during execution of trigger 'ADMIN.CHECKSURRENDERREASON'

Error at Line: 7 Column: 0

2:28:43 AM - Code execution finished

Powered by: ODS

11°C Clear

Search

9:28 PM 3/27/2024

3] Creating a trigger to ensure that the date of acquisition of an animal is always after the date of birth of the animal.

Autonomous Database | Oracle | SQL | Oracle Database Actions | +

g769cec454f4d27-ohq7zn97f4mq1ww.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/_sdw/?nav=worksheet

ORACLE Database Actions | SQL

Search Database (Ctrl+K)

ADMIN

Navigator

Files

ADMIN

Tables

Search...

ACCOMMODATE

ADOPTION_DETAILS

ANIMAL_FEATURES

ANIMAL_INFO

ANIMAL_TRANSFER_RECORDS

CONTAIN

DONE_BY

EMPLOYEES_STAFF_DETAILS

EMPLOYEE_PAYROLL

EXAMINE

FACILITIES

GROOMING_HISTORY

MEDICAL_HISTORY

NEED

NUTRITION

PAYMENT_DETAILS

PERFORM

EMPLOYEE_PAYROLL

GetFundsForShelter*

Consumer group: LOW

Data Load

```
1 CREATE OR REPLACE TRIGGER CheckDOB
2 BEFORE INSERT ON animal_info
3 FOR EACH ROW
4 BEGIN
5     IF :NEW.dob > :NEW.date_of_acquisition THEN
6         RAISE_APPLICATION_ERROR(-20001, 'Date of birth cannot be greater than date of acquisition.');
```

Query Result

Script Output

DBMS Output

Explain Plan

Autotrace

SQL History

Trigger CHECKDOB compiled

Elapsed: 00:00:00.022

ORA-04088: error during execution of trigger 'ADMIN.CHECKSURRENDERREASON'

Error at Line: 7 Column: 0

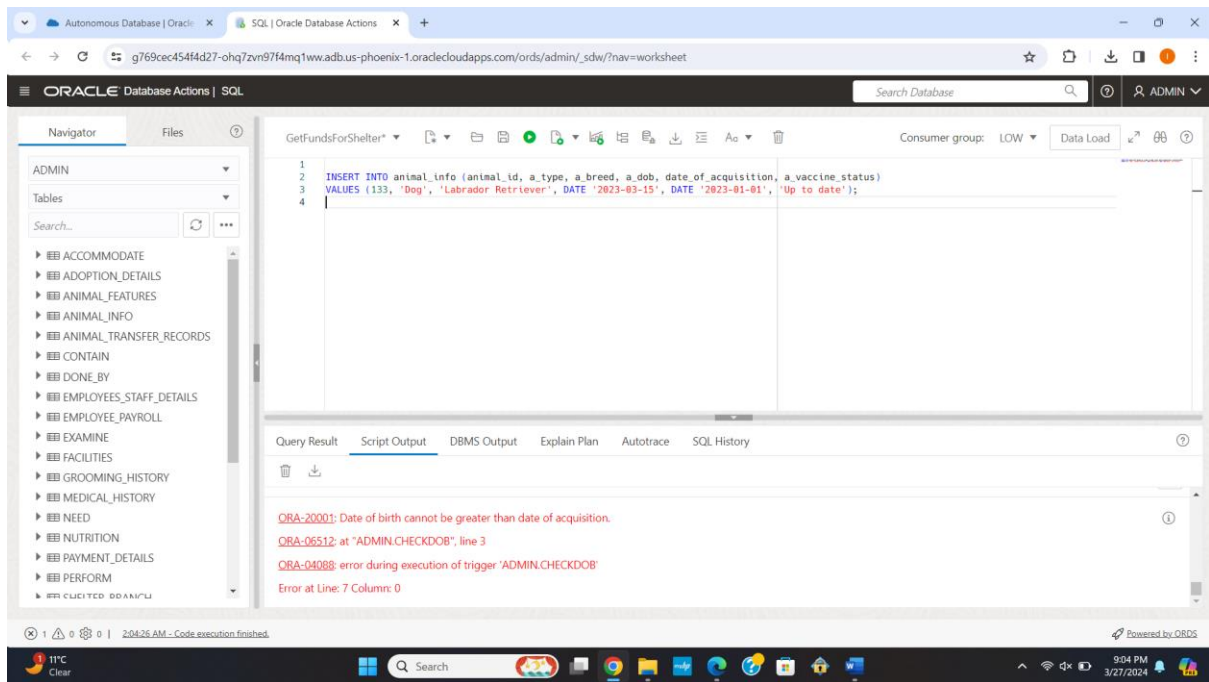
2:03:49 AM - SQL executed by ADMIN

Powered by: ODS

11°C Clear

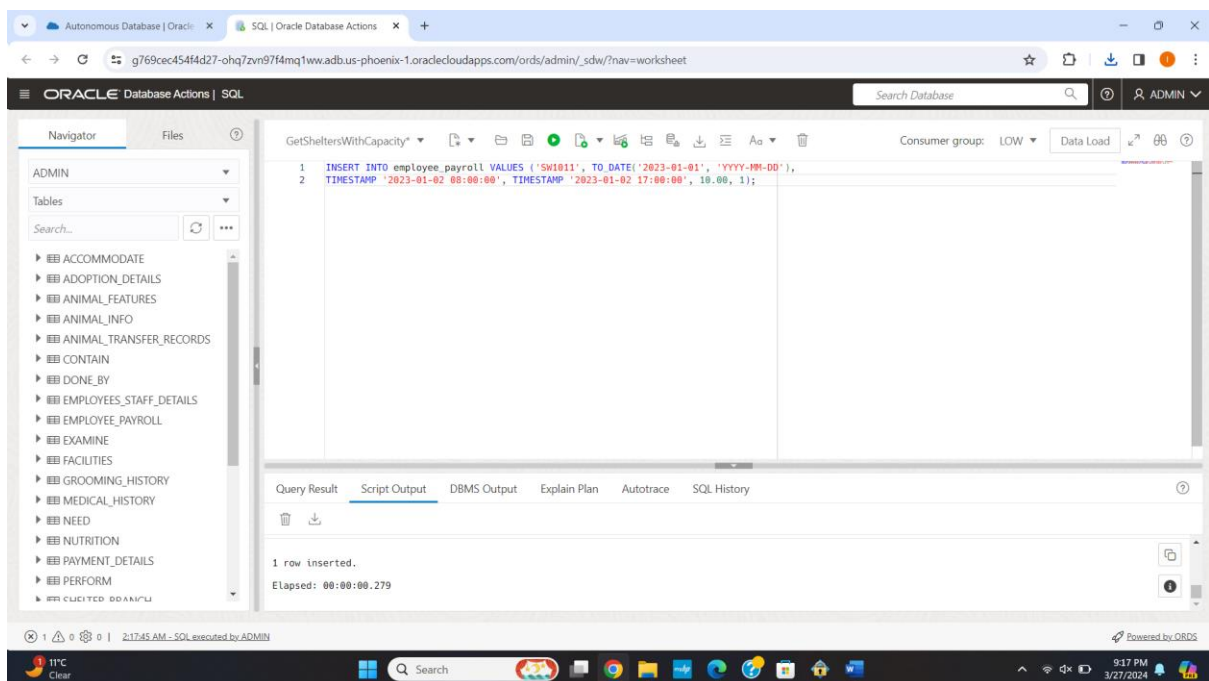
Search

9:03 PM 3/27/2024



Package

1] Creating a package to calculate the total pay and taxes considering the number of hours worked.



Autonomous Database | Oracle SQL | Oracle Database Actions

g769cec454fd27-ohq7zvn97f4mq1ww.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/_sdw/?nav=worksheet

ORACLE Database Actions | SQL

Search Database

ADMIN

Navigator

Files

Employee_Salary_Package*

Consumer group: LOW

Data Load

```
1 CREATE OR REPLACE PACKAGE Employee_Salary_Package AS
2   PROCEDURE calculate_gross_salary(p_emp_id IN INT, p_gross_salary OUT NUMBER);
3   FUNCTION calculate_total_taxes(p_gross_salary IN NUMBER) RETURN NUMBER;
4   FUNCTION calculate_net_salary(p_gross_salary IN NUMBER, p_taxes IN NUMBER) RETURN NUMBER;
5 END Employee_Salary_Package;
```

Query Result Script Output DBMS Output Explain Plan Autotrace SQL History

Elapsed: 00:00:00.279

Package EMPLOYEE_SALARY_PACKAGE compiled

Elapsed: 00:00:00.026

2:18:55 AM - SQL executed by ADMIN

Powered by ODS

11°C Clear

Search

9:19 PM 3/27/2024

Autonomous Database | Oracle SQL | Oracle Database Actions

g769cec454fd27-ohq7zvn97f4mq1ww.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/_sdw/?nav=worksheet

ORACLE Database Actions | SQL

Search Database (Ctrl+K)

ADMIN

Navigator

Files

Employee_Salary_Package*

Consumer group: LOW

Data Load

```
1 CREATE OR REPLACE PACKAGE BODY Employee_Salary_Package AS
2   PROCEDURE calculate_gross_salary(p_emp_id IN INT, p_gross_salary OUT NUMBER) AS
3     v_gross_salary NUMBER := 0;
4     v_swipe_total NUMBER;
5     v_swipe_id VARCHAR2(100);
6   BEGIN
7     FOR swipe_record IN (SELECT swipe_id FROM employee_payroll WHERE emp_id = p_emp_id) LOOP
8       v_swipe_id := swipe_record.swipe_id;
9       SELECT (EXTRACT(HOUR FROM (check_out_time - check_in_time)) +
10        EXTRACT(MINUTE FROM (check_out_time - check_in_time)) / 60) * hourly_pay
11        INTO v_swipe_total
12        FROM employee_payroll
13        WHERE swipe_id = v_swipe_id;
14       v_gross_salary := v_gross_salary + v_swipe_total;
15     END LOOP;
16     p_gross_salary := v_gross_salary;
17   END calculate_gross_salary;
18   FUNCTION calculate_total_taxes(p_gross_salary IN NUMBER) RETURN NUMBER AS
19     v_total_taxes NUMBER := 0;
20   BEGIN
21     IF p_gross_salary > 100 THEN
22       v_total_taxes := p_gross_salary * 0.1; -- 10% tax
23     END IF;
24     RETURN v_total_taxes;
25   END calculate_total_taxes;
```

Query Result Script Output DBMS Output Explain Plan Autotrace SQL History

2:20:24 AM - SQL executed by ADMIN

Powered by ODS

11°C Clear

Search

9:20 PM 3/27/2024

Autonomous Database | Oracle | SQL | Oracle Database Actions

g769cec454f4d27-ohq7zn97f4mq1ww.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/_sdw/?nav=worksheet

ORACLE Database Actions | SQL

Search Database

ADMIN

Navigator

Files

Employee_Salary_Package*

Consumer group: LOW

Data Load

```
29 END calculate_total_taxes;
30
31 FUNCTION calculate_net_salary(p_gross_salary IN NUMBER, p_taxes IN NUMBER) RETURN NUMBER AS
32   v_net_salary NUMBER;
33 BEGIN
34   v_net_salary := p_gross_salary - p_taxes;
35   RETURN v_net_salary;
36 END calculate_net_salary;
37 END Employee_Salary_Package;
38 /
39
```

Query Result

Script Output

DBMS Output

Explain Plan

Autotrace

SQL History

1 row inserted.

Elapsed: 00:00:00.279

Package EMPLOYEE_SALARY_PACKAGE compiled

Elapsed: 00:00:00.026

Package Body EMPLOYEE_SALARY_PACKAGE compiled

Elapsed: 00:00:00.036

2:20:24 AM - SQL executed by ADMIN

Powered by: ODS

11°C Clear

Search

9:21 PM 3/27/2024

Cloud Sign In | SQL | Oracle Database Actions

g769cec454f4d27-ohq7zn97f4mq1ww.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/_sdw/?nav=worksheet

ORACLE Database Actions | SQL

Search Database

ADMIN

Navigator

Files

Employee_Salary_Package*

Consumer group: LOW

Data Load

```
1 DECLARE
2   p_emp_id INT := 1;
3   v_gross_salary NUMBER;
4   v_total_taxes NUMBER;
5   v_net_salary NUMBER;
6 BEGIN
7   Employee_Salary_Package.calculate_gross_salary(p_emp_id, v_gross_salary);
8   v_total_taxes := Employee_Salary_Package.calculate_total_taxes(v_gross_salary);
9   v_net_salary := Employee_Salary_Package.calculate_net_salary(v_gross_salary, v_total_taxes);
10
11   DBMS_OUTPUT.PUT_LINE('Pay check details for Employee ID ' || p_emp_id || ':');
12   DBMS_OUTPUT.PUT_LINE('Gross salary: ' || TO_CHAR(v_gross_salary, '99999.99'));
13   DBMS_OUTPUT.PUT_LINE('Total taxes: ' || TO_CHAR(v_total_taxes, '99999.99'));
14   DBMS_OUTPUT.PUT_LINE('Net salary: ' || TO_CHAR(v_net_salary, '99999.99'));
15 END;
16 /
17
18
19
```

Query Result

Script Output

DBMS Output

Explain Plan

Autotrace

SQL History

Pay check details for Employee ID 1:

Gross salary: 188.00

Total taxes: 18.00

Net salary: 162.00

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.046

2:31:53 AM - SQL executed by ADMIN

Powered by: ODS

11°C Clear

Search

9:32 PM 3/27/2024

Individual Contribution:

The intention of this part was to develop an electronic environment that would help the shelter very purpose to save the animals and the personnel looking after them, rather than developing just a database.

This is the part where the 'GetFundsForShelter' function has been created. In line with being an important part of making sure financial accountability and transparency are adhered to, I have designed it in a manner where it can perfectly keep track and systematically record all amounts received for each shelter under its sponsors. This was a way to ensure that each donation went straight into being able to better the life of the animals rather than a way of fundraising. Accordingly, the function was written correctly, tested rigorously, and used by others, including this, to become a reliable source in maintaining and helping to create a supportive and trusting environment of personnel and contributors of the shelter.

I developed the 'GetSheltersWithCapacity' Procedure focusing on the operating features of the shelters. To be precise, it aims at enabling a simplified yet effective way of the improved resources' allocation, operational planning, and identified shelters that over a certain limit barrier therein. It is my real hope that this information will be able to create a true bridge between data administration and practical parts of shelter life, so that every shelter will have a possibility of working within its means while still maintaining the highest standards of animal care.

The creation of the 'CheckDOB' trigger was perhaps the most meaningful example of my commitment to the part. Realization made on the above shortcoming instigated the creation of this trigger to ensure the integrity of the animals' records, especially the date of acquisition being logically sequenced after the date of birth, given the basic significance of correct record-keeping in animal welfare. This feature of the database system does show a deep respect for the travels and pasts of these animals to ensure their care began on a foundation of truth and trust. It confirmed for me that I can use technology to improve and safeguard the lives of those under our care. Each trigger, procedure, and function developed was goal-centric and commitment had "commitment" as a word for seeing measurable improvement in the shelter community.