Animal Shelter Database System Project

Project Objective:

Organizing tons of data in every organization is important to effectively manage operations and streamline processes. QUELIFE is an animal shelter organization for which we will be designing a database system. This organization has multiple branches spread across Wisconsin, Illinois, Missouri, Montana, California, Texas, and Nevada. This database will encompass primary details related to facilities, animal descriptions and transfer details, sponsor details and employee details. We are also ensuring to maintain high quality of the database using normalization.

Requirement Analysis:

We are analysing the requirements to setup a database for this organization. We have multiple entities to consider along with their relationships. Here we have multiple branches of shelter at multiple locations with various facilities. Animal related records including animal information, animal features, adoption records and surrender details are need as well as the history of each animal. We are enabling the option to transfer of animals from one branch to another. Employee details, sponsor details and payment details are required to track the working of the shelter. Applying normalization to the tables and splitting the tables to ensure that there is no information loss.

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
22. Employee department is selected depending on their education only
23. Employee has a unique hourly pay until unless the pay is changed and the table is updated
24. Each breed determines the type of animal
25. Address of the shelter has an unique pincode

Updated Entities and Relationships:

1. **shelter_branch:**
    - Entity Description: This entity contains details about the geographic location and details of each branch
    - Attributes: shelter_name, shelter_loc, address, shelter_contact
    - Relationships:
        - One-to-one relationship with facilities
        - Many-to-many relationship with animal_info
        - One-to-many relationship with employees_staff_details
        - Many-to-many relationship with animal_transfer_records
        - Many-to-many relationship with sponsor
        - Many-to-one relationship with shelter_loc_details

2. **shelter_loc_details (new table)**
    - Entity Description: This entity contains details about the shelter location of shelter
    - Attributes: shelter_loc, pin_code
    - Relationships:
        - One-to-many relationship with shelter_branch

3. **facilities:**
    - Entity Description: This entity represents amenities operated by each branch.
    - Attributes: fid, f_capacity, f_ani_count, f_area, play_areas, temp_control, outdoor_space
    - Relationships:
        - One-to-one relationship with shelter_branch

4. **animal_info:**
    - Entity Description: This entity contains information about the animal.
    - Attributes: animal_id, a_breed, a_dob ,date_of_acquisition, a_vaccine_status, animal_features_id
    - Relationships:
        - Many-to-many relationship with shelter_branch (as we have transfer facility)
        - One-to-one relationship with animal_features
        - Many-to-many relationship with surrender_details
        - Many-to-many relationship with adoption_details
        - One-to-many relationship with animal_transfer_records
        - Many-to-many relationship with nutrition
        - Ternary relationship with history of the animal which includes: grooming history, training history and medical history
        - Many-to-one relationship with *animal_breed_info*
5. ***animal_breed_info (new table)***
    - Entity Description: This entity contains information about the animal breed.
    - Attributes: a_breed, a_type
    - Relationships:
        - One-to-many relationship with animal_info
6. **animal_features:**
    - Entity Description: This entity contains physical attributes of the animal.
    - Attributes: animal_id, a_color, a_weight, a_height, a_eye_color, a_condition

- Relationships:
    - One-to-one relationship with animal_info

7. **adoption_details:**
    - Entity Description: This entity contains information about the adoption of an animal
    - Attributes: ad_ID, animal_id, ad_name, ad_contact, ad_date, ad_fee.
    - Relationships:
        - Many-to-many relationship with animal_info
        - One-to-one relationship with payment_details

8. **surrender_details:**
    - Entity Description: This entity contains details about returning of the animal to the shelter
    - Attributes: s_id, s_date, s_name (person who brings animals), s_reason.
    - Relationships:
        - Many-to-many relationship with animal_info

9. **nutrition:**
    - Entity Description: This contains information about diet and nutrition of each animal
    - Attributes: food_id, food_type, food_quantity
    - Relationships:
        - Many-to-many relationship with animal_info

10. **training_history:**
    - Entity Description: This entity contains details about training the animals
    - Attributes: training_ID, trainer_name, training_date, training_type, training_stage.
    - Relationships:
        - Many-to-one relationship with animal_info
        - Many-to-many relationship with employees_staff_details

11. **grooming_history:**
    - Entity Description: This entity contains details about grooming the animals
    - Attributes: grooming_id, groomer_name, grooming_date, grooming_type.
    - Relationships:
        - Many-to-one relationship with animal_info
        - Many-to-many relationship with employees_staff_details

12. **medical_history:**
    - Entity Description: This entity contains details about the medical history of each animal
    - Attributes: medical_id, disease, treatment_date
    - Relationships:
        - Many-to-one relationship with animal_info
        - Many-to-many relationship with vet_id
        - One-to-one relationship between medical_history and vaccination
        - Many-to-one relationship with disease_mitigation

13. **disease_mitigation** ***(new table)***
   - Entity Description: This entity contains details about the disease_mitigation of each animal.
   - Attributes: disease, vaccination_id
   - Relationships:
     - One-to-many relationship with medical_history
     - One-to-one relationship between disease_mitigation and vaccination

14. **vaccination:**
   - Entity Description: This entity contains details about the vaccination of each animal
   - Attributes: vaccination_id, vaccination_name, vaccination_date, vaccination_dosage
   - Relationships:
     - One-to-one relationship between medical_history and vaccination

15. **employees_staff_details:**
   - Entity Description: This entity contains details about each worker or employee at the animal shelter QUELIFE
   - Attributes: emp_id, emp_name, emp_contact, emp_training, emp_education, emp_criminal_record
   - Relationships:
     - Many-to-one relationship with shelter_branch
     - Many-to-many relationship with grooming_history
     - Many-to-many relationship with training_history
     - Unary relationship – manages – branch manager
     - Many-to-one relationship with employees_stage

16. **employees_stage** ***(new table)***
   - Entity Description: This entity contains details about each employee stage at the animal shelter QUELIFE
   - Attributes: emp_level , emp_department, emp_education
   - Relationships:
     - One-to-many relationship with employees_staff_details

17. **sponsor:**
   - Entity Description: This entity contains details about sponsors given to shelter_branch
   - Attributes: sponsor_id, sponsor_name, sponsor_contact, sponsorship_type.
   - Relationships:
     - Many-to-many relationship with shelter_branch
     - One-to-many relationship with payment_details

18. **payment_details:**
   - Attributes: payment_ID, amount, payment_Date, payment_type.
   - Relationships:
     - One-to-one relationship with adoption_details
     - Many-to-one relationship with sponsor

19. **veterinary:**

- Entity Description: This entity contains details of the veterinarian
- Attributes: vet_id, vet_name, vet_contact, education
- Relationships:
    - Many-to-many relationship with medical_history
    - Many-to-one relationship with vet_edu

20. **Vet_edu*(new table):***
    - Entity Description: This entity contains details of the veterinarian education
    - Attributes: vet_name, vet_contact, work_exp
    - Relationships:
        - One-to-many relationship with veterinary
21. **animal_transfer_records:**
    - Entity Description: This entity contains details of transfer of animals from one branch to another
    - Attributes: transfer_id, transfer_date, transfer_from, transfer_to.
    - Relationships:
        - Many-to-one relationship with animal_info
        - Many-to-many relationship with shelter_branch
22. **employee_payroll:**
    - Entity Description: This entity contains details of the working hours of each employee.
    - Attributes: swipe_id, swipe_date, check_in_time , check_out_time, emp_id
    - Relationships:
        - Many-to-one relationship with employees_staff_details
        - Many-to-one relationship with employee_pay
23. **employee_pay*(new table):***
    - Entity Description: This entity contains details of the hourly pay of each employee.
    - Attributes: emp_id, hourly_pay
    - Relationships:
        - One-to-many relationship with employee_pay
        - Many-to-one relationship with employees_staff_details

UPDATED SCHEMA: **(Primary Key: Underlined, Foreign Key: Italics)**

shelter_branch (shelter_id, shelter_name, shelter_loc, address, shelter_contact)

animal_info (animal_id , a_breed, a_dob, date_of_acquisition, a_vaccine_status, *animal_features_id*
)

animal_features (animal_features_id, a_color, a_weight, a_height, a_eye_color, a_condition)

animal_transfer_records (transfer_id, transfer_date, transfer_from, transfer_to, *animal_id*)

grooming_history (grooming_id, groomer_name, grooming_date, grooming_type)

nutrition (food_id, food_type, food_quantity)

sponsor (sponsor_id, sponsor_name, sponsor_contact, sponsorship_type)

surrender_details (s_id, s_date, s_name, s_reason)

training_history (training_ID, trainer_name, training_date, training_type, training_stage)

veterinary (vet_id, vet_name, vet_contact, education)

vet_specialization(*vet_id*, specialization)

medical_history (medical_id, disease, treatment_date)

payment_details (payment_ID, *sponsor_id* , amount, payment_Date, payment_type)

employees_staff_details (emp_id, emp_name, emp_contact, emp_training, emp_education, emp_criminal_record)

facilities (facilities_id, *shelter_id* , f_capacity, f_ani_count, f_area, play_areas, temp_control, outdoor_space)

adoption_details (ad_ID, *payment_ID*, ad_name, ad_contact, ad_date, ad_fee)

vaccination (vaccination_id, vaccination_name, vaccination_date, vaccination_dosage)

employee_payroll (swipe_id, *emp_id*, swipe_date, check_in_time, check_out_time)

perform *(grooming_Id, emp_id)*

examines *(vet_id, medical_id)*

support *(shelter_id, sponsor_id)*

contain (*animal_id, ad_ID*)

need (*food_id, animal_id*)

submit (*s_id, animal_id)*

accommodate (*shelter_id, animal_id)*

transfers (*shelter_id, transfer_id)*

done_by *(training_ID, emp_id)*

stores (animal_id, training_history_id, grooming_history_id, medical_history) shelter_loc_details (shelter_loc, pin_code)

animal_breed_info (a_breed, a_type)

disease_mitigation (disease, vaccination_id)

employees_stage(emp_level, emp_department, emp_education)

vet_edu (vet_name, vet_contact, work_exp)

employee_pay (*emp_id*, hourly_pay)

Database Normalization (Part 6):

Database normalization is necessary to design high quality tables. This step is necessary to maintain redundancy of the tables in such a way that no dependant information is lost due to deletion of any data rows. The normalize the tables in the database it is necessary to identify the functional dependencies in each table. Here is a list of functional dependencies we have identified in each table:

1. **shelter_branch:**
   shelter_id -> shelter_name, shelter_loc (shelter_id uniquely determines shelter_name and shelter_loc)

   shelter_id, shelter_name -> address, shelter_contact (For each tuple of shelter_ID and shelter_name, there is a unique address and contact)

   shelter_loc -> pin_code (shelter_loc uniquely determines pin_code)

2. **facilities:**
   facilities_id -> f_capacity, f_ani_count, temp_control, outdoor_space, shelter_id (shelter_id uniquely determines shelter_name and shelter_loc)

   facilities_id, outdoor_space -> f_area, play_areas

3. **animal_info:**
   animal_id -> a_type, a_breed, a_dob, date_of_acquisition, a_vaccine_status
   animal_id, a_type -> animal_features_id
   a_breed -> a_type

4. **animal_features:**
   animal_features_id -> a_color, a_weight, a_height, a_condition, a_eye_color
   animal_features_id, a_condition -> a_color, a_weight, a_height

5. **adoption_details:**
   ad_ID -> ad_name, ad_contact, ad_date, ad_fee, payment_ID
   ad_ID, ad_date -> payment_ID, ad_name, ad_contact

6. **surrender_details:**
   s_id -> s_date, s_name

s_id, s_date -> s_reason

7. **nutrition:**
   food_id -> food_type
   food_id, food_type -> food_quantity

8. **training_history:**
   training_ID -> trainer_name, training_date, training_type
   training_ID, training_date -> training_stage

9. **grooming_history:**
   grooming_id -> groomer_name, grooming_date
   grooming_id, groomer_name -> grooming_type

10. **medical_history:**
    medical_id -> disease, vaccination_id
    medical_id, disease -> vaccination_id, treatment_date

11. **vaccination:**
    vaccination_id -> vaccination_name, vaccination_date
    vaccination_id, vaccination_name -> vaccination_dosage

12. **employees_staff_details:**
    emp_id -> emp_name, emp_contact, emp_training, emp_education, emp_criminal_record
    emp_id, emp_contact -> emp_criminal_record
    emp_education -> emp_level
    emp_education -> emp_department

13. **sponsor:**
    sponsor_id -> sponsor_name, sponsor_contact
    sponsor_id, sponsor_name, sponsor_contact -> sponsorship_type

14. **payment_details:**
    payment_ID -> sponsor_id, payment_Date, payment_type
    payment_ID, sponsor_id -> amount, payment_Date, payment_type

15. **veterinary:**
    vet_id -> vet_name, vet_contact, education, work_exp
    vet_name, vet_contact -> work_exp

16. **animal_transfer_records:**
    transfer_id -> transfer_date, animal_id
    transfer_id, animal_id -> transfer_date, transfer_from, transfer_to

17. **employee_payroll:**
    swipe_id -> swipe_date, check_in_time, check_out_time, emp_id
    emp_id -> hourly_pay

Considering the FDs mentioned above, we are now checking if the tables are normalized. To normalize database system into BCNF preserving dependencies and maintaining lossless join, we are analysing each table and then decomposing into new tables wherever necessary. We can say that a schema or table is in BCNF if all the functional dependencies for that table

1. **shelter_branch:**
   shelter_id -> shelter_name, shelter_loc:
   Proof:
   This functional dependency satisfies the requirements of BCNF as the determinant (LHS: shelter_id) is the primary key which implies it is a super key.
   shelter_id+ = {shelter_id, shelter_name, shelter_loc, pincode, shelter_contact }
   As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

   shelter_id, shelter_name -> address, shelter_contact:
   Proof:
   This functional dependency satisfies the requirements of BCNF as the determinant (LHS: shelter_id, shelter_name) is a super key as we can determine all other attributes using these 2.
   Proof:
   (shelter_id, shelter_name)+ = { shelter_id, shelter_name, shelter_loc, pincode, shelter_contact }
   As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

   shelter_loc -> pin code: This functional dependency does not satisfy the requirements of BCNF as the determinant (LHS: shelter_loc) is not a super key and therefore there is a need of a new table. Shelter location cannot be used to determine each row in the table.

   So, we are splitting this table into 2:
   shelter_branch: (old table)
   shelter_id
   shelter_name
   shelter_loc
   address
   shelter_contact

   shelter_loc_details (new table)
   shelter_loc
   pin_code

   Proof for preservation of Functional Dependencies post-split:
   All the functional dependencies either fall into the old or new table created
   shelter_id -> shelter_name, shelter_loc: This dependency is preserved in the old table
   shelter_id, shelter_name -> address, shelter_contact: This dependency is preserved in the old table
   shelter_loc -> pin_code: This dependency is preserved in the new table

   Thus both tables are in BCNF

   Proof for Lossless Joins:

To ensure that the join is lossless either of the conditions mentioned below should be satisfied:
R1 ∩ R2 -> R1 – R2 or R1 ∩ R2 -> R2 – R1 (where R1 is the old table and R2 is the new table)
R1 ∩ R2 = { shelter_loc }

R1 ∩ R2 -> R1 – R2 = shelter_loc -> {shelter_id, shelter_name, shelter_loc , address, shelter_contact}
R1 ∩ R2 -> R2 – R1 = shelter_loc -> pin_code

Second condition is satisfied and thus this join is said to be lossless.

2. **facilities:**
facilities_id -> f_capacity, f_ani_count, temp_control, outdoor_space, shelter_id:
Proof:
This functional dependency satisfies the requirements of BCNF as the determinant (LHS: facilities_id) is the primary key which implies it is a super key.
facilities_id + = { facilities_id, shelter_id, f_capacity, f_ani_count, f_area, play_areas, temp_control, outdoor_space }
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

facilities_id, outdoor_space -> f_area, play_areas: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: facilities_id, outdoor_space) is a super key as we can determine all other attributes using these 2.
(facilities_id, outdoor_space) + = { facilities_id, shelter_id, f_capacity, f_ani_count, f_area, play_areas, temp_control, outdoor_space }
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

This table satisfies conditions of BCNF.

3. **animal_info:**
animal_id -> a_type, a_breed, a_dob, date_of_acquisition, a_vaccine_status: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: animal_id) is the primary key which implies it is a super key.
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

animal_id, a_type -> animal_features_id: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: animal_id, a_type) is a super key as we can determine all other attributes using these 2.
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

a_breed -> a_type: This functional dependency does not satisfy the requirements of BCNF as the determinant (LHS: a_breed) is not a super key and therefore there is a need of a new table. Breed of animal cannot be used to determine other attributes in the table.
So, we are splitting this table into 2:
animal_info: (old table)
animal_id

a_breed
a_dob
date_of_acquisition
a_vaccine_status
animal_features_id
animal_breed_info (new table)
a_breed
a_type

Proof for preservation of Functional Dependencies post-split:
All the functional dependencies either fall into the old or new table created
animal_id -> a_type, a_breed, a_dob, date_of_acquisition, a_vaccine_status: This dependency is preserved in the old table
animal_id, a_type -> animal_features_id: This dependency is preserved in the old table
a_breed -> a_type: This dependency is preserved in the new table

Thus, both tables are in BCNF

Proof for Lossless Joins:
To ensure that the join is lossless either of the conditions mentioned below should be satisfied:
R1 ∩ R2 -> R1 – R2 or R1 ∩ R2 -> R2 – R1 (where R1 is the old table and R2 is the new table)
R1 ∩ R2 = { a_breed }

R1 ∩ R2 -> R1 – R2 = a_breed -> { animal_id, a_dob, date_of_acquisition, a_vaccine_status, animal_features_id}
R1 ∩ R2 -> R2 – R1 = a_breed -> a_type

Second condition is satisfied and thus this join is said to be lossless.

4. **animal_features:**
animal_features_id -> a_color, a_weight, a_height, a_condition, a_eye_color: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: animal_features_id) is primary key which implies it is a super key.
{ animal_features_id+ = {animal_features_id, a_color, a_weight, a_height, a_eye_color, a_condition}
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

animal_features_id, a_condition -> a_color, a_weight, a_height: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: animal_features_id, a_condition) is a super key as animal_feature_id and a_condition can be used to get the other details in other attributes.
(animal_features_id, a_condition)+ ={ animal_features_id, a_condition, a_color, a_weight, a_height, a_eye_color}
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

5. **adoption_details:**

ad_ID -> ad_name, ad_contact, ad_date, ad_fee, payment_ID : This functional dependency satisfies the requirements of BCNF as the determinant (LHS: ad_ID) is the primary key which implies it is a super key.
ad_ID+ = { ad_ID, payment_ID, ad_name, ad_contact, ad_date, ad_fee}
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

ad_ID, ad_date -> payment_ID, ad_name, ad_contact: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: ad_ID, ad_date) is a super key as ad_ID and ad_can be used to get the other details in other attributes.
ad_ID, ad_date+ ={ ad_ID, ad_date payment_ID, ad_name, ad_contact, ad_fee}
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

6. **surrender_details:**
   s_id -> s_date, s_name: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: s_id) is the primary key which implies it is a super key.
   s_id+ ={s_id, s_date, s_name, s_reason}
   As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

   s_id, s_date -> s_reason: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: s_id, s_date) is a super key as s_id and s_date can be used to get the other details in other attributes.
   s_id, s_date+ ={ s_id, s_date, s_name, s_reason}
   As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

7. **nutrition:**
   food_id -> food_type: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: food_id) is the primary key which implies it is a super key.
   food_id+ ={ food_id, food_type, food_quantity}
   As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

   food_id, food_type -> food_quantity: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: food_id, food_type) is a super key as food_id and food_type can be used to get the other details in other attributes.
   (food_id, food_type)+ ={ food_id, food_type, food_quantity}
   As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

8. **training_history:**
   training_ID -> trainer_name, training_date, training_type: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: training_ID) is the primary key which implies it is a super key.

training_ID+ ={ training_ID, trainer_name, training_date, training_type, training_stage}
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

training_ID, training_date -> training_stage: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: training_ID, training_date) is a super key as training_ID and training_date can be used to get the other details in other attributes.
training_ID, training_date+ ={ training_ID, training_date, trainer_name, training_type, training_stage}
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

9. **grooming_history:**
grooming_id -> groomer_name, grooming_date: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: grooming_id) is the primary key which implies it is a super key.
grooming_id+ ={ grooming_id, groomer_name, grooming_date, grooming_type}
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

grooming_id, groomer_name -> grooming_type: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: grooming_id, groomer_name) is a super key as grooming_id and groomer_name can be used to get the other details in other attributes.
grooming_id, groomer_name+={ grooming_id, groomer_name, grooming_date, grooming_type}
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

10. **medical_history:**
medical_id -> disease, vaccination_id: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: medical_id) is the primary key which implies it is a super key.
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

medical_id, disease -> vaccination_id, treatment_date: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: medical_id, disease) is a super key as we can determine all other attributes using these on LHS.
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

disease -> vaccination_id: This functional dependency does not satisfy the requirements of BCNF as the determinant (LHS: disease) is not a super key and therefore there is a need of a new table. Disease name cannot be used to determine each row in the table.
So, we are splitting this table into 2:
medical_history: (old table)
medical_id
disease

treatment_date
disease_mitigation (new table)
disease
vaccination_id

Proof for preservation of Functional Dependencies post-split:
All the functional dependencies either fall into the old or new table created
medical_id -> disease, vaccination_id: : This dependency is preserved in the old table
medical_id, disease -> vaccination_id, treatment_date: : This dependency is preserved in the old table
disease -> vaccination_id: : This dependency is preserved in the new table

Thus, both tables are in BCNF

Proof for Lossless Joins:
To ensure that the join is lossless either of the conditions mentioned below should be satisfied:
R1 ∩ R2 -> R1 – R2 or R1 ∩ R2 -> R2 – R1 (where R1 is the old table and R2 is the new table)
R1 ∩ R2 = { a_breed }

R1 ∩ R2 -> R1 – R2 = disease -> { medical_id, treatment_date}
R1 ∩ R2 -> R2 – R1 = disease -> vaccination_id

Second condition is satisfied and thus this join is said to be lossless.

11. **vaccination:**
vaccination_id -> vaccination_name, vaccination_date: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: vaccination_id) is the primary key which implies it is a super key.
vaccination_id+ = {vaccination_id, vaccination_name, vaccination_date, vaccination_dosage }
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

vaccination_id, vaccination_name -> vaccination_dosage: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: vaccination_id, vaccination_name) is a super key as vaccination_id and vaccination_name can be used to get the other details in other attributes.
vaccination_id, vaccination_name + ={ vaccination_id, vaccination_name, vaccination_date, vaccination_dosage}
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

12. **employees_staff_details:**
emp_id -> emp_name, emp_contact, emp_training, emp_education, emp_criminal_record: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: emp_id) is the primary key which implies it is a super key.
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

emp_id, emp_contact -> emp_criminal_record: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: emp_id, emp_contact) is a super key as we can determine all other attributes using these on LHS.
As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

emp_education -> emp_level: This functional dependency does not satisfy the requirements of BCNF as the determinant (LHS: emp_education) is not a super key and therefore there is a need of a new table. emp_education name cannot be used to determine each row in the table.
emp_education -> emp_department: This functional dependency does not satisfy the requirements of BCNF as the determinant (LHS: emp_education) is not a super key and therefore there is a need of a new table. emp_education name cannot be used to determine each row in the table.

So, we are splitting this table into 2:
employees_staff_details***: (old table)***
emp_id
emp_name
emp_contact
emp_training
emp_education
emp_criminal_record
employees_stage *(new table)*
emp_education
emp_department
emp_level

Proof for preservation of Functional Dependencies post-split:
All the functional dependencies either fall into the old or new table created
emp_id -> emp_name, emp_contact, emp_training, emp_education, emp_criminal_record: This dependency is preserved in the old table
emp_id, emp_contact -> emp_criminal_record: This dependency is preserved in the old table
emp_education -> emp_level: This dependency is preserved in the new table
emp_education -> emp_department: This dependency is preserved in the new table

Thus, both tables are in BCNF

Proof for Lossless Joins:
To ensure that the join is lossless either of the conditions mentioned below should be satisfied:
R1 ∩ R2 -> R1 – R2 or R1 ∩ R2 -> R2 – R1 (where R1 is the old table and R2 is the new table)
R1 ∩ R2 = { emp_education }

R1 ∩ R2 -> R1 – R2 = emp_education -> { emp_id, emp_name, emp_contact, emp_training , emp_criminal_record}
R1 ∩ R2 -> R2 – R1 = emp_education -> emp_department, emp_level

Second condition is satisfied and thus this join is said to be lossless with the combination of third and fourth FDs.

13. **sponsor:**

    sponsor_id -> sponsor_name, sponsor_contact: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: sponsor_id) is the primary key which implies it is a super key.

    As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

    sponsor_id, sponsor_name, sponsor_contact -> sponsorship_type: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: sponsor_id, sponsor_name, sponsor_contact) is a super key as sponsor_id, sponsor_name and sponsor_contact can be used to get the other details in other attributes.
    As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

14. **payment_details:**

    payment_ID -> sponsor_id, payment_Date, payment_type: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: payment_ID) is the primary key which implies it is a super key.
    As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

    payment_ID, sponsor_id -> amount, payment_Date, payment_type: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: payment_ID, sponsor_id) is a super key as payment_ID and sponsor_id can be used to get the other details in other attributes.
    As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

15. **veterinary:**

    vet_id -> vet_name, vet_contact, education, work_exp: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: vet_id) is the primary key which implies it is a super key.
    As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

    vet_name, vet_contact -> work_exp: This functional dependency does not satisfy the requirements of BCNF as the determinant (LHS: vet_name, vet_contact) is not a super key and therefore there is a need of a new table. vet_name and vet_contact cannot be used to determine each row in the table.

    So, we are splitting this table into 2:
    veterinary*: (old table)*

vet_id
vet_name
vet_contact
education
vet_edu *(new table)*
vet_name
vet_contact
work_exp

Proof for preservation of Functional Dependencies post-split:
All the functional dependencies either fall into the old or new table created
emp_education -> emp_department: This dependency is preserved in the new table
vet_id -> vet_name, vet_contact, education, work_exp: This dependency is preserved in the old table
vet_name, vet_contact -> work_exp: This dependency is preserved in the new table

Thus, both tables are in BCNF

Proof for Lossless Joins:
To ensure that the join is lossless either of the conditions mentioned below should be satisfied:
R1 ∩ R2 -> R1 – R2 or R1 ∩ R2 -> R2 – R1 (where R1 is the old table and R2 is the new table)
R1 ∩ R2 = { vet_name, vet_contact }

R1 ∩ R2 -> R1 – R2 = vet_name, vet_contact -> { vet_id, education}
R1 ∩ R2 -> R2 – R1 = vet_name, vet_contact -> work_exp

Second condition is satisfied and thus this join is said to be lossless with the combination of third and fourth FDs.

16. **animal_transfer_records:**
transfer_id -> transfer_date, animal_id: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: transfer_id) is the primary key which implies it is a super key. As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

transfer_id, animal_id -> transfer_date, transfer_from, transfer_to: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: transfer_id, animal_id) is a super key as transfer_id, animal_id can be used to get the other details in other attributes. As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

17. **employee_payroll:**
swipe_id -> swipe_date, check_in_time, check_out_time, emp_id: This functional dependency satisfies the requirements of BCNF as the determinant (LHS: swipe_id) is the primary key which implies it is a super key.

As we can get the information of all attributes from the attribute mentioned on left side of the relation, we can say that this FD satisfies BCNF.

emp_id -> hourly_pay: This functional dependency does not satisfy the requirements of BCNF as the determinant (LHS: emp_id) is not a super key and therefore there is a need of a new table. emp_id name cannot be used to determine each row in the table as there might be multiple swipes for each employee id.
So, we are splitting this table into 3:
employee_payroll*: (old table)*
swipe_id
swipe_date
check_in_time
check_out_time
emp_id
employee_pay*(new table)*
emp_id
hourly_pay

Proof for preservation of Functional Dependencies post-split:
All the functional dependencies either fall into the old or new table created
swipe_id -> swipe_date, check_in_time, check_out_time, emp_id: This dependency is preserved in the old table
emp_id -> hourly_pay: This dependency is preserved in the new table

Thus, both tables are in BCNF

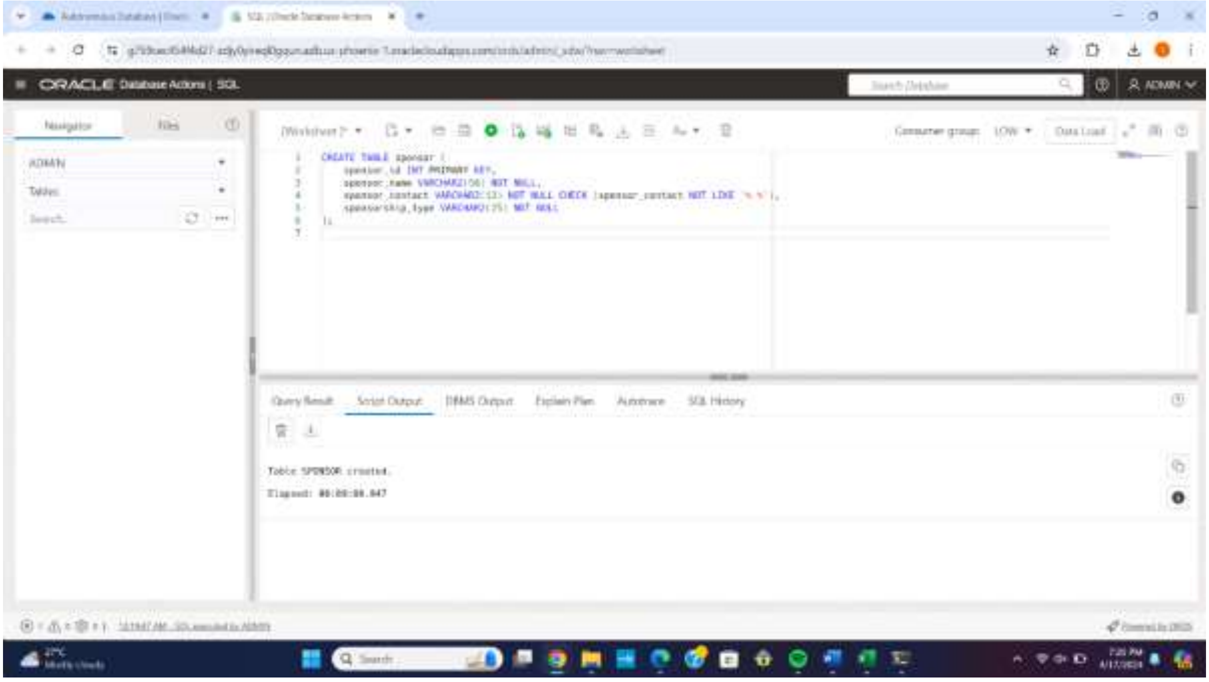
Proof for Lossless Joins:
To ensure that the join is lossless either of the conditions mentioned below should be satisfied:
R1 ∩ R2 -> R1 – R2 or R1 ∩ R2 -> R2 – R1 (where R1 is the old table and R2 is the new table)
R1 ∩ R2 = { emp_id }

R1 ∩ R2 -> R1 – R2 = emp_id -> { swipe_id, swipe_date, check_in_time, check_out_time }
R1 ∩ R2 -> R2 – R1 = emp_id -> hourly_pay

Second condition is satisfied and thus this join is said to be lossless with the combination of third and fourth FDs.

Updated Database Screenshots:

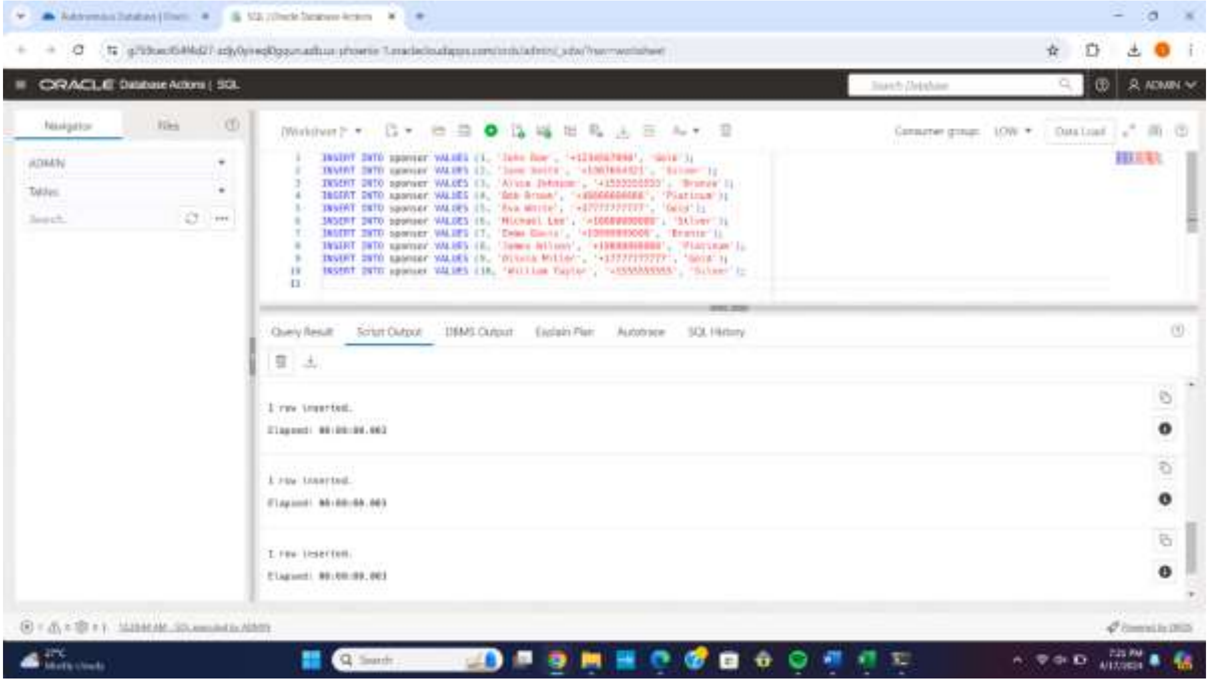
Creating a table named sponsor



```
CREATE TABLE sponsor (
    sponsor_id INT PRIMARY KEY,
    sponsor_name VARCHAR2(50) NOT NULL,
    sponsor_contact VARCHAR2(13) NOT NULL CHECK (sponsor_contact NOT LIKE '% %'),
    sponsorship_type VARCHAR2(25) NOT NULL
);
```

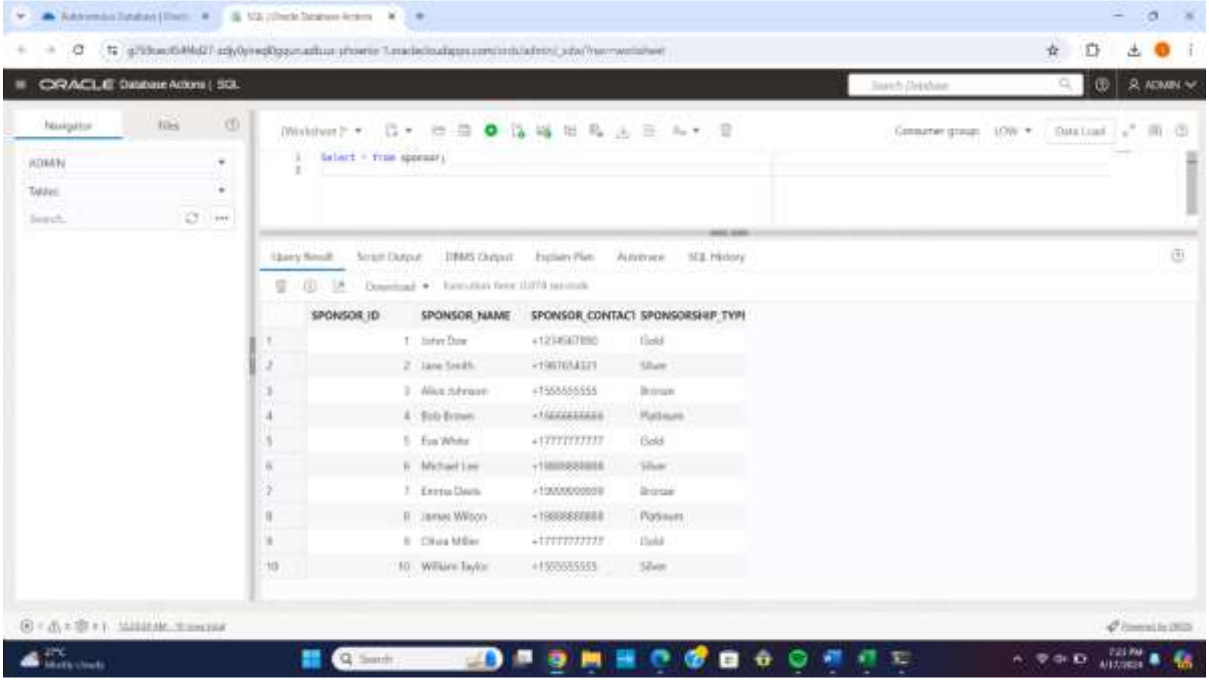
```
Table SPONSOR created.

Elapsed: 00:00:00.047
```

Inserting values to sponsor



```
INSERT INTO sponsor VALUES (1, 'John Doe', '+1234567890', 'Gold');
INSERT INTO sponsor VALUES (2, 'Jane Smith', '+1987654321', 'Silver');
INSERT INTO sponsor VALUES (3, 'Alice Johnson', '+1333333333', 'Bronze');
INSERT INTO sponsor VALUES (4, 'Bob Brown', '+1666666666', 'Platinum');
INSERT INTO sponsor VALUES (5, 'Eva White', '+1777777777', 'Gold');
INSERT INTO sponsor VALUES (6, 'Michael Lee', '+1000000000', 'Silver');
INSERT INTO sponsor VALUES (7, 'Emma Davis', '+1999999999', 'Bronze');
INSERT INTO sponsor VALUES (8, 'James Wilson', '+1888888888', 'Platinum');
INSERT INTO sponsor VALUES (9, 'Olivia Miller', '+1777777777', 'Gold');
INSERT INTO sponsor VALUES (10, 'William Taylor', '+1555555555', 'Silver');
```

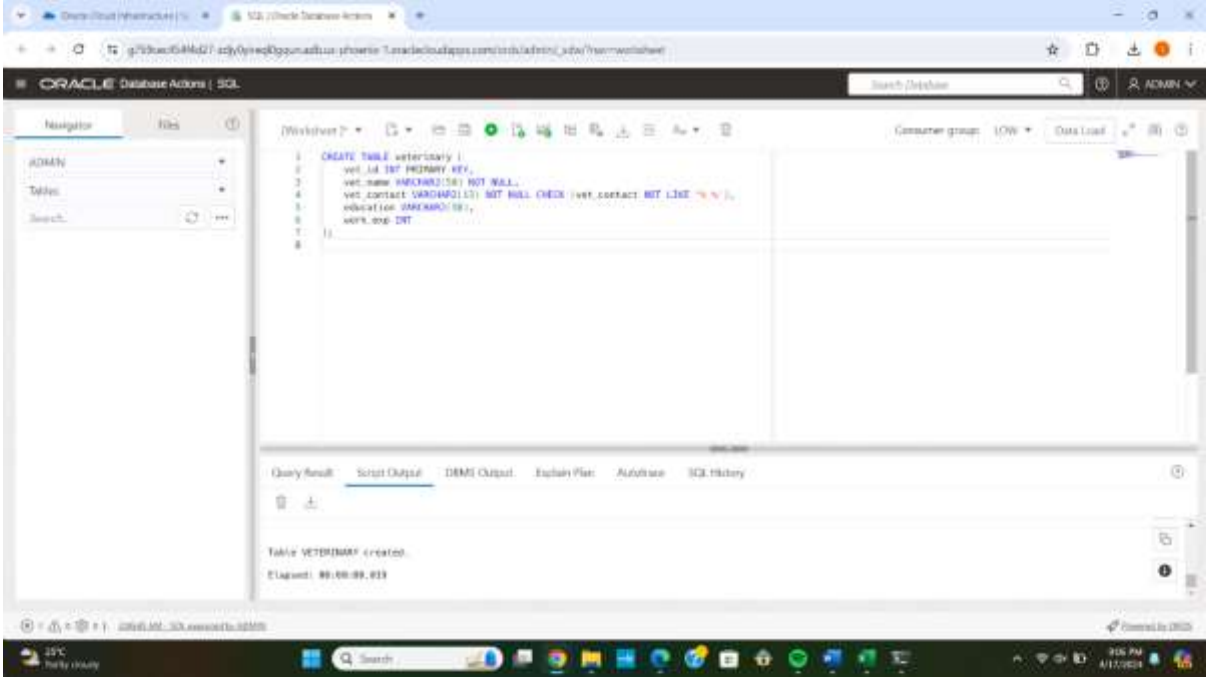
```
1 row inserted.

Elapsed: 00:00:00.003

1 row inserted.

Elapsed: 00:00:00.003

1 row inserted.

Elapsed: 00:00:00.003
```



```
Select * from sponsor;
```

	SPONSOR_ID	SPONSOR_NAME	SPONSOR_CONTACT	SPONSORSHIP_TYPI
1	1	John Doe	+1234567890	Gold
2	2	Jane Smith	+1987654321	Silver
3	3	Alice Johnson	+1555555555	Bronze
4	4	Bob Brown	+1666666666	Platinum
5	5	Eva White	+1777777777	Gold
6	6	Michael Lee	+1888888888	Silver
7	7	Emma Davis	+1999999999	Bronze
8	8	James Wilson	+1888888888	Platinum
9	9	Olivia Miller	+1777777777	Gold
10	10	William Taylor	+1555555555	Silver

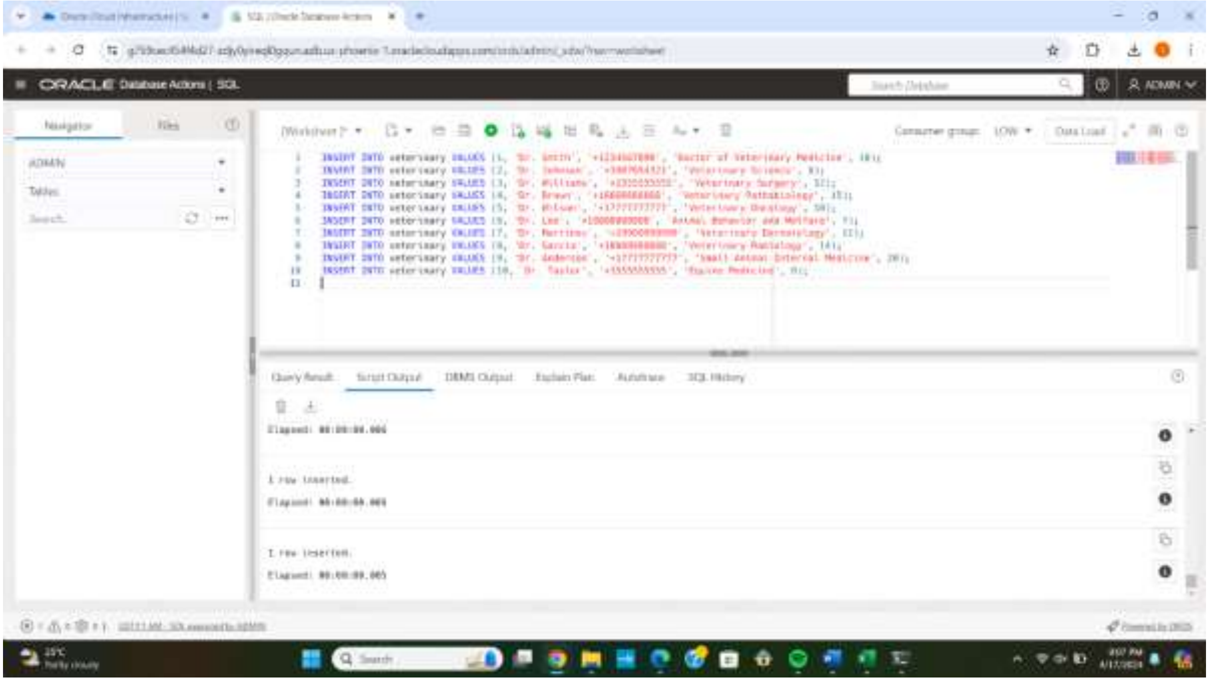
Creating a table named veterinary



```
CREATE TABLE veterinary (
    vet_id INT PRIMARY KEY,
    vet_name VARCHAR2(50) NOT NULL,
    vet_contact VARCHAR2(13) NOT NULL CHECK (vet_contact NOT LIKE '% %'),
    education VARCHAR2(50),
    work_exp INT
);
```

```
Table VETERINARY created.

Elapsed: 00:00:00.019
```

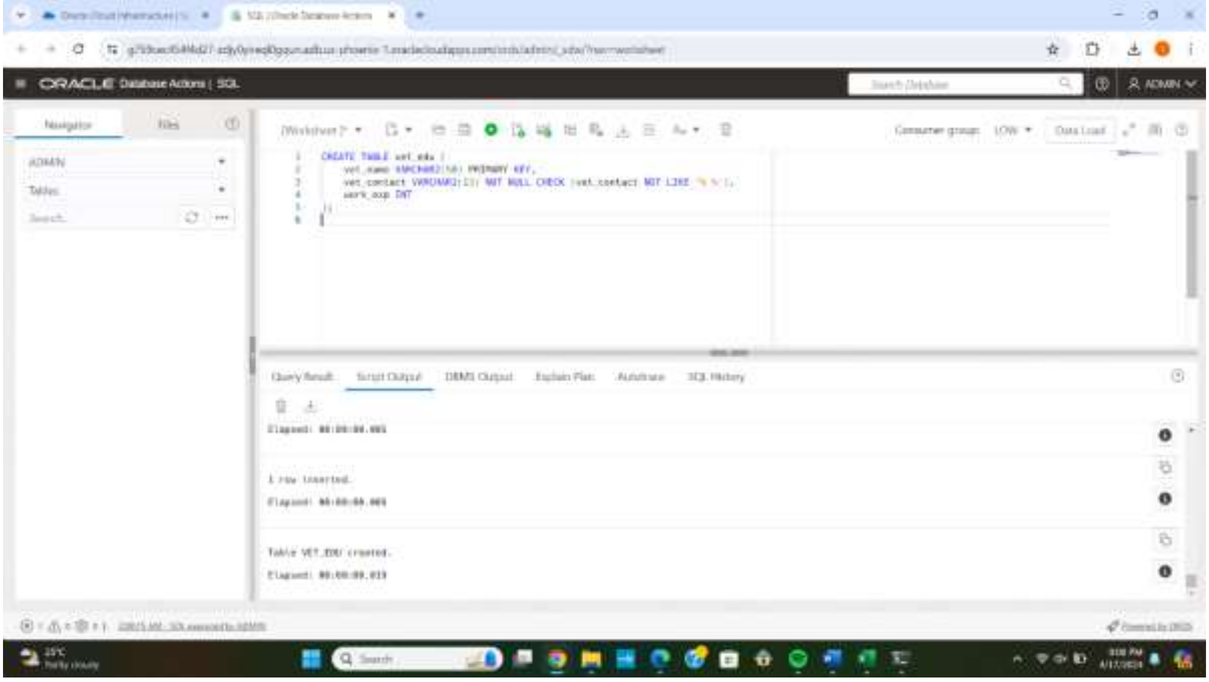
Inserting values to veterinary



```
INSERT INTO veterinary VALUES (1, 'Dr. Smith', '+1234567890', 'Doctor of Veterinary Medicine', 10);
INSERT INTO veterinary VALUES (2, 'Dr. Johnson', '+1987654321', 'Veterinary Science', 8);
INSERT INTO veterinary VALUES (3, 'Dr. Williams', '+1555555555', 'Veterinary Surgery', 12);
INSERT INTO veterinary VALUES (4, 'Dr. Brown', '+1666666666', 'Veterinary Pathobiology', 15);
INSERT INTO veterinary VALUES (5, 'Dr. Wilson', '+1777777777', 'Veterinary Oncology', 20);
INSERT INTO veterinary VALUES (6, 'Dr. Lee', '+1000000000', 'Animal Behavior and Welfare', 7);
INSERT INTO veterinary VALUES (7, 'Dr. Martinez', '+1990000000', 'Veterinary Dermatology', 11);
INSERT INTO veterinary VALUES (8, 'Dr. Garcia', '+1888888888', 'Veterinary Radiology', 14);
INSERT INTO veterinary VALUES (9, 'Dr. Anderson', '+1777777777', 'Small Animal Internal Medicine', 20);
INSERT INTO veterinary VALUES (10, 'Dr. Taylor', '+1555555555', 'Equine Medicine', 9);
```

```
Elapsed: 00:00:00.006

1 row inserted.
Elapsed: 00:00:00.003

1 row inserted.
Elapsed: 00:00:00.005
```

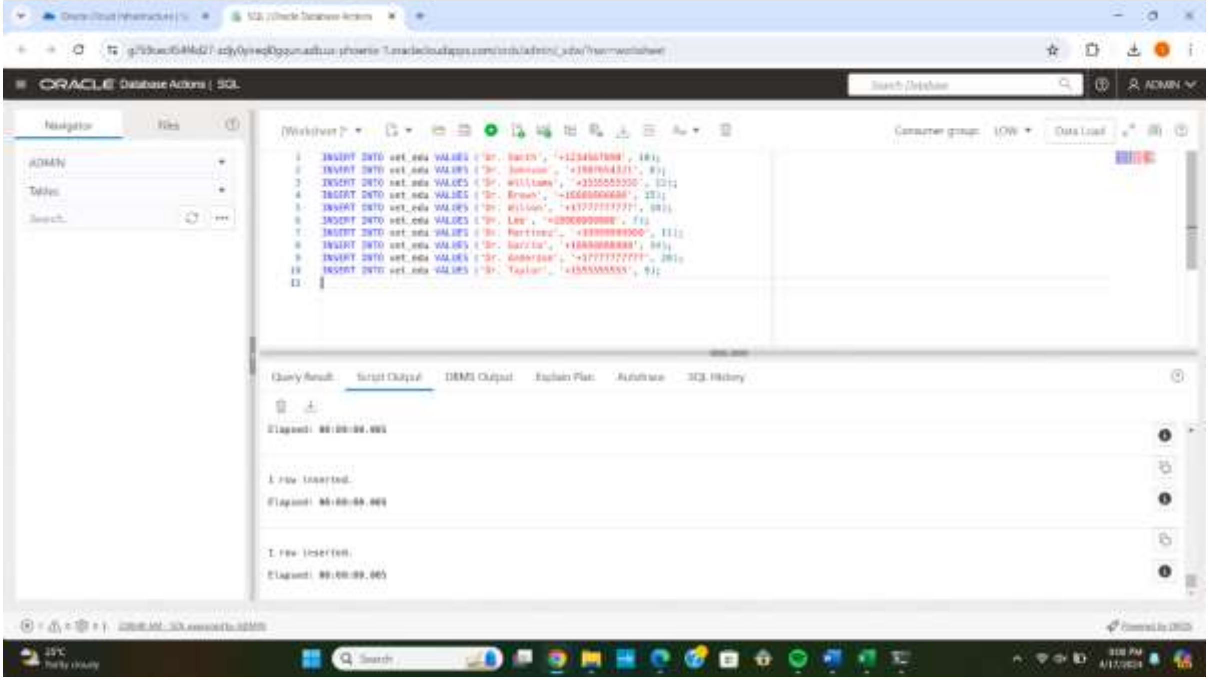
Creating a table named vet_edu(New)



```
CREATE TABLE vet_edu (
    vet_name VARCHAR2(50) PRIMARY KEY,
    vet_contact VARCHAR2(15) NOT NULL CHECK (vet_contact NOT LIKE '%-%'),
    work_exp INT
);
```

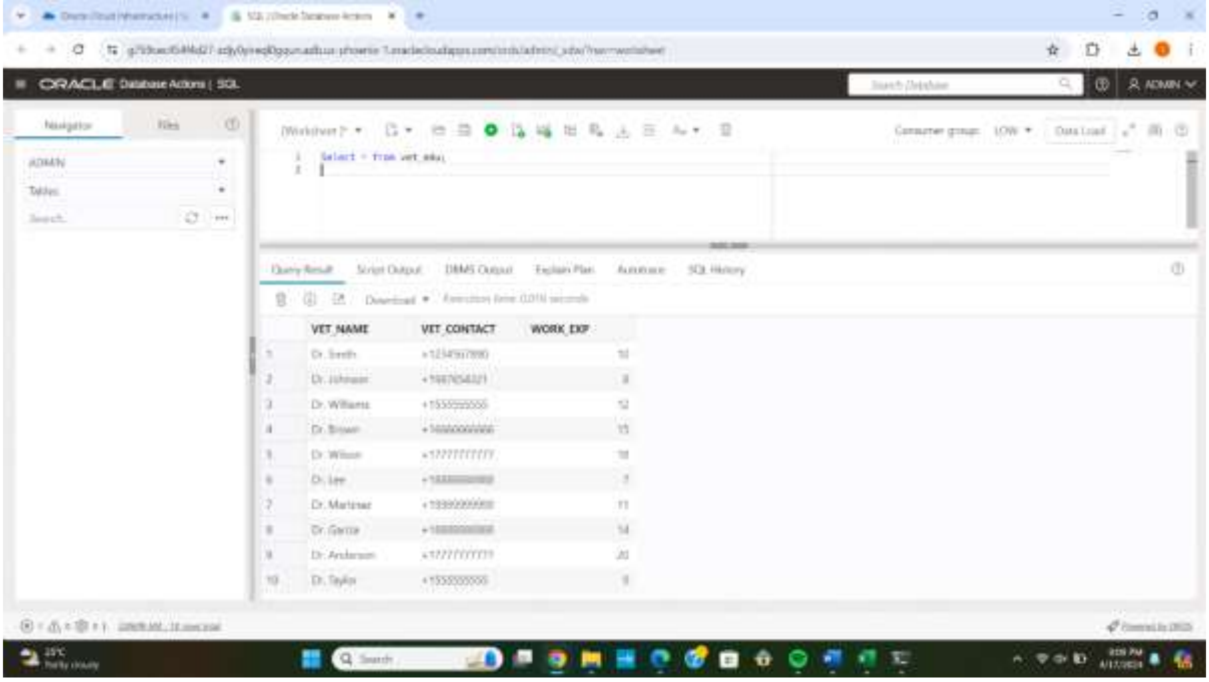
```
Elapsed: 00:00:00.005

1 row inserted.
Elapsed: 00:00:00.003

Table VET_EDU created.
Elapsed: 00:00:00.019
```

Inserting values to vet_edu(New)



```
INSERT INTO vet_edu VALUES ('Dr. Smith', '+1234567890', 10);
INSERT INTO vet_edu VALUES ('Dr. Johnson', '+1987654321', 8);
INSERT INTO vet_edu VALUES ('Dr. Williams', '+1555555555', 12);
INSERT INTO vet_edu VALUES ('Dr. Brown', '+1666666666', 15);
INSERT INTO vet_edu VALUES ('Dr. Wilson', '+17777777777', 18);
INSERT INTO vet_edu VALUES ('Dr. Lee', '+1888888888', 7);
INSERT INTO vet_edu VALUES ('Dr. Martinez', '+1999999999', 11);
INSERT INTO vet_edu VALUES ('Dr. Garcia', '+1888888888', 14);
INSERT INTO vet_edu VALUES ('Dr. Anderson', '+17777777777', 20);
INSERT INTO vet_edu VALUES ('Dr. Taylor', '+1555555555', 9);
```

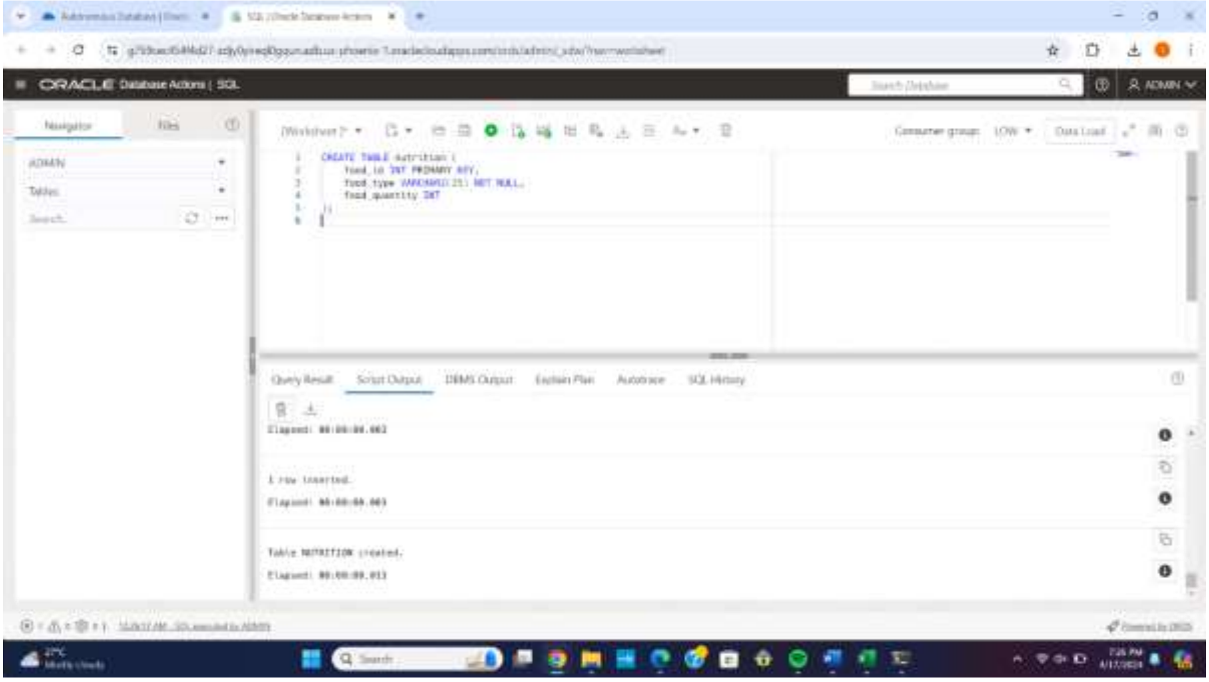
```
Elapsed: 00:00:00.005

1 row inserted.

Elapsed: 00:00:00.005

1 row inserted.

Elapsed: 00:00:00.005
```



```
Select * from vet_edu;
```

	VET_NAME	VET_CONTACT	WORK_EXP
1	Dr. Smith	+1234567890	10
2	Dr. Johnson	+1987654321	8
3	Dr. Williams	+1555555555	12
4	Dr. Brown	+1666666666	15
5	Dr. Wilson	+17777777777	18
6	Dr. Lee	+1888888888	7
7	Dr. Martinez	+1999999999	11
8	Dr. Garcia	+1888888888	14
9	Dr. Anderson	+17777777777	20
10	Dr. Taylor	+1555555555	9

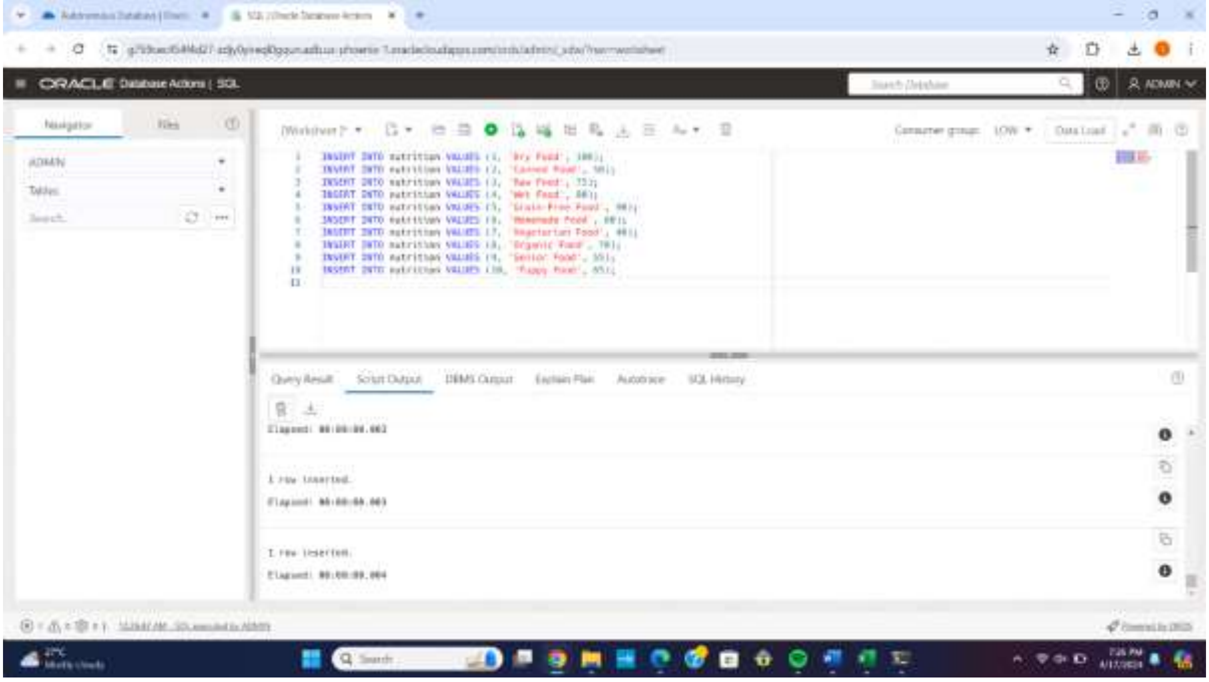
Creating a table named nutrition



```
CREATE TABLE nutrition (
    food_id INT PRIMARY KEY,
    food_type VARCHAR2(25) NOT NULL,
    food_quantity INT
);
```

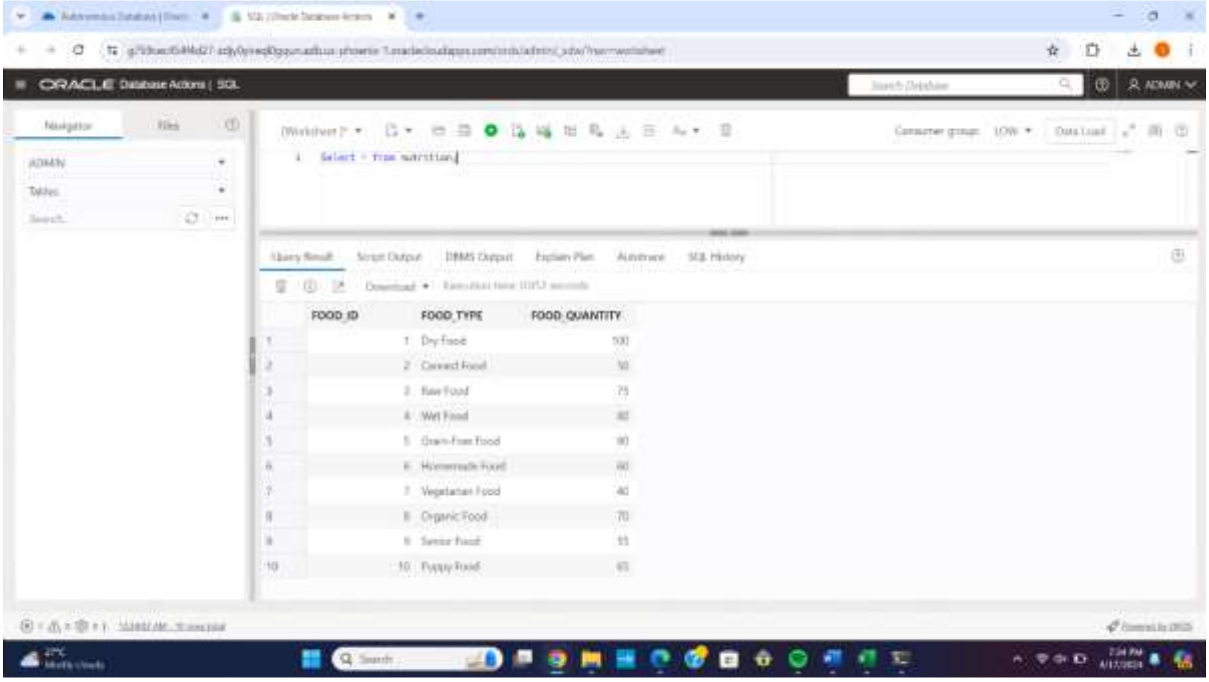
```
Elapsed: 00:00:00.002

1 row inserted.

Elapsed: 00:00:00.003

Table NUTRITION created.

Elapsed: 00:00:00.013
```

Inserting values to nutrition



```
INSERT INTO nutrition VALUES (1, 'Dry Food', 100);
INSERT INTO nutrition VALUES (2, 'Canned Food', 50);
INSERT INTO nutrition VALUES (3, 'Raw Food', 75);
INSERT INTO nutrition VALUES (4, 'Wet Food', 80);
INSERT INTO nutrition VALUES (5, 'Grain-Free Food', 90);
INSERT INTO nutrition VALUES (6, 'Homemade Food', 60);
INSERT INTO nutrition VALUES (7, 'Vegetarian Food', 40);
INSERT INTO nutrition VALUES (8, 'Organic Food', 70);
INSERT INTO nutrition VALUES (9, 'Senior Food', 55);
INSERT INTO nutrition VALUES (10, 'Puppy Food', 65);
```

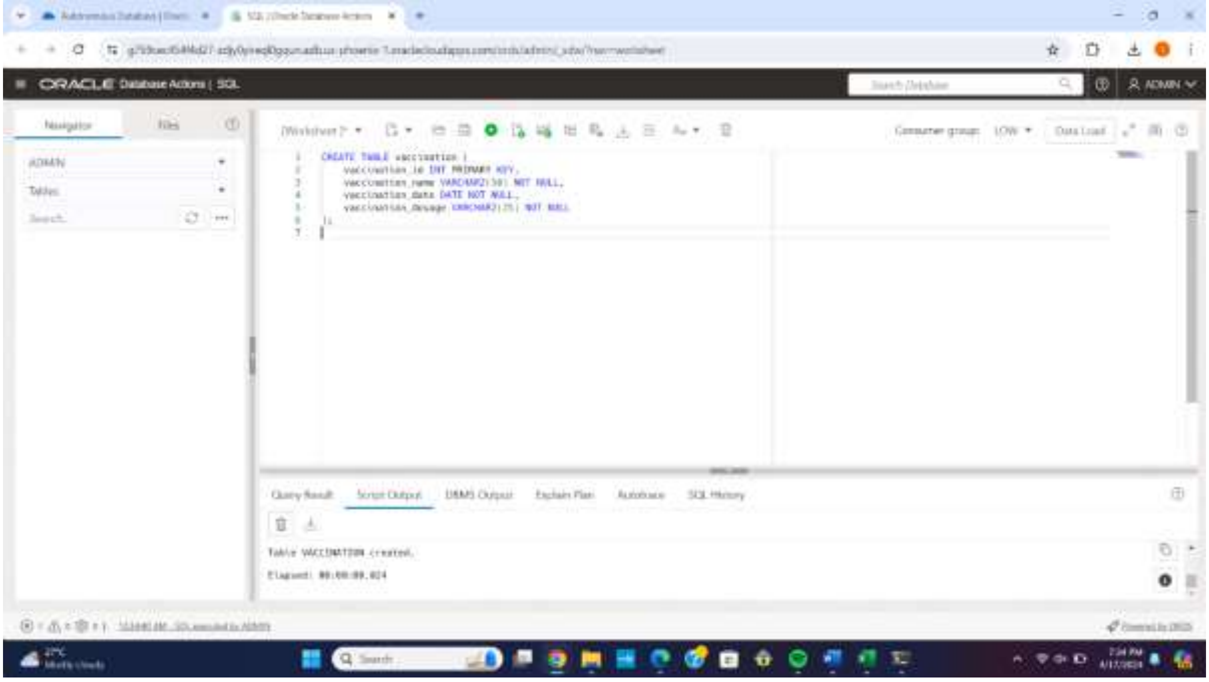
```
Elapsed: 00:00:00.002

1 row inserted.

Elapsed: 00:00:00.003

1 row inserted.

Elapsed: 00:00:00.004
```



```
Select * from nutrition;
```

	FOOD_ID	FOOD_TYPE	FOOD_QUANTITY
1	1	Dry Food	100
2	2	Canned Food	50
3	3	Raw Food	75
4	4	Wet Food	80
5	5	Grain-Free Food	90
6	6	Homemade Food	60
7	7	Vegetarian Food	40
8	8	Organic Food	70
9	9	Senior Food	55
10	10	Puppy Food	65

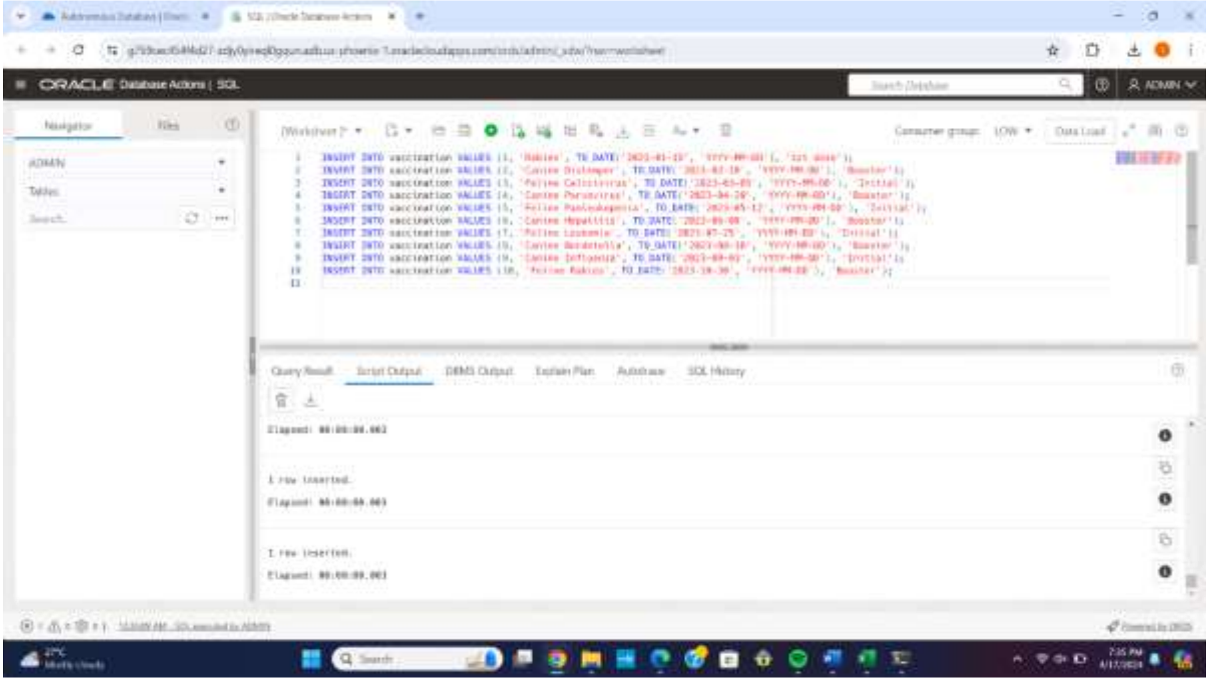
Creating a table named vaccination



```
CREATE TABLE vaccination (
    vaccination_id INT PRIMARY KEY,
    vaccination_name VARCHAR2(30) NOT NULL,
    vaccination_date DATE NOT NULL,
    vaccination_dosage VARCHAR2(25) NOT NULL
);
```

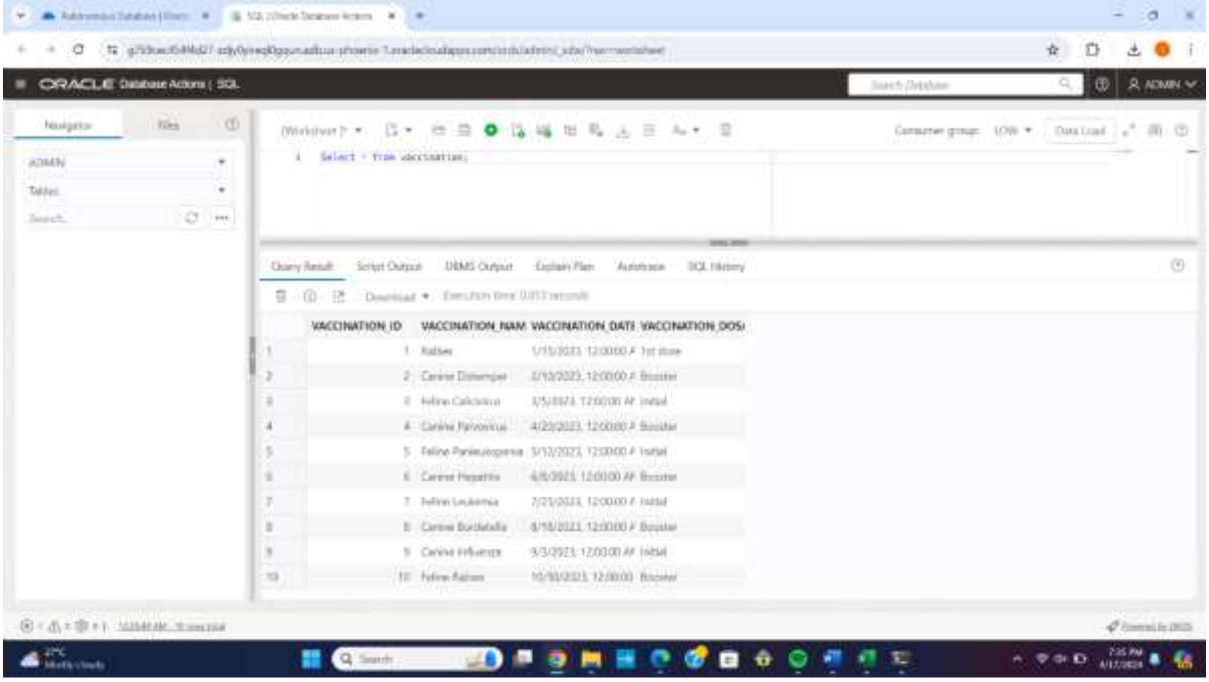
```
Table VACCINATION created.

Elapsed: 00:00:00.024
```

Inserting values to vaccination



```
INSERT INTO vaccination VALUES (1, 'Rabies', TO_DATE('2023-01-15', 'YYYY-MM-DD'), '1st dose');
INSERT INTO vaccination VALUES (2, 'Canine Distemper', TO_DATE('2023-02-10', 'YYYY-MM-DD'), 'Booster');
INSERT INTO vaccination VALUES (3, 'Feline Calicivirus', TO_DATE('2023-03-05', 'YYYY-MM-DD'), 'Initial');
INSERT INTO vaccination VALUES (4, 'Canine Parvovirus', TO_DATE('2023-04-20', 'YYYY-MM-DD'), 'Booster');
INSERT INTO vaccination VALUES (5, 'Feline Panleukopenia', TO_DATE('2023-05-12', 'YYYY-MM-DD'), 'Initial');
INSERT INTO vaccination VALUES (6, 'Canine Hepatitis', TO_DATE('2023-06-08', 'YYYY-MM-DD'), 'Booster');
INSERT INTO vaccination VALUES (7, 'Feline Leukemia', TO_DATE('2023-07-25', 'YYYY-MM-DD'), 'Initial');
INSERT INTO vaccination VALUES (8, 'Canine Bordetella', TO_DATE('2023-08-18', 'YYYY-MM-DD'), 'Booster');
INSERT INTO vaccination VALUES (9, 'Canine Influenza', TO_DATE('2023-09-03', 'YYYY-MM-DD'), 'Initial');
INSERT INTO vaccination VALUES (10, 'Feline Rabies', TO_DATE('2023-10-30', 'YYYY-MM-DD'), 'Booster');
```

```
Elapsed: 00:00:00.003

1 row inserted.

Elapsed: 00:00:00.003

1 row inserted.

Elapsed: 00:00:00.003
```

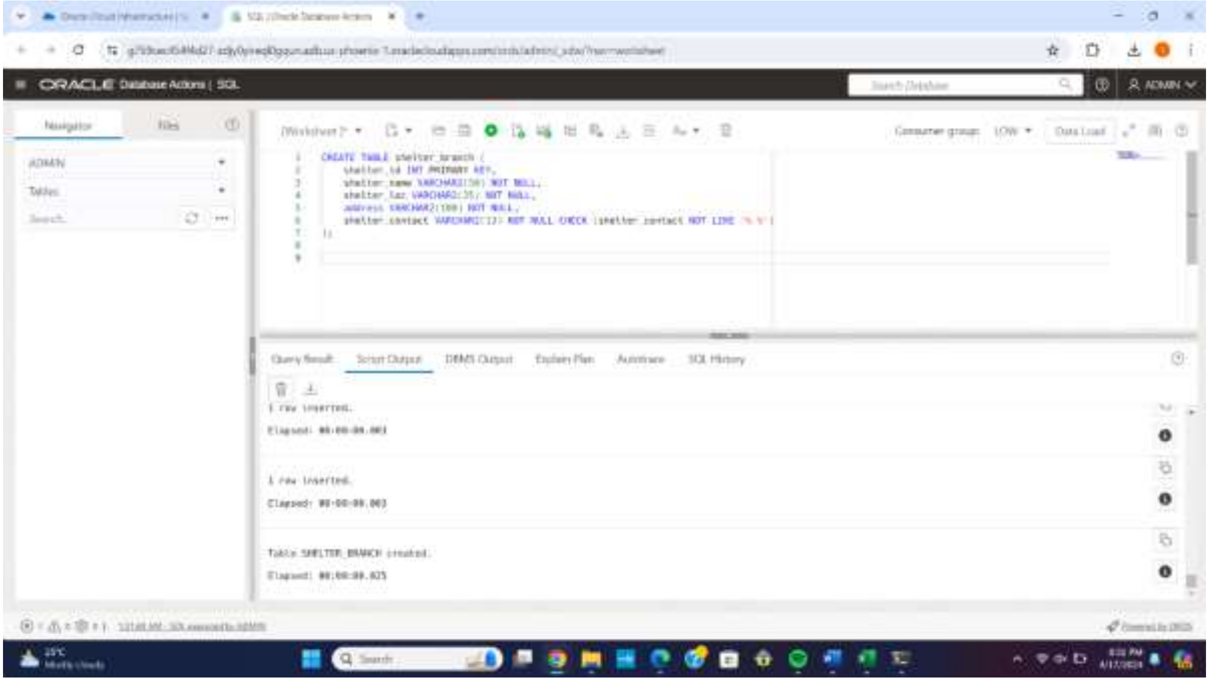


```
Select * from vaccination;
```

Execution time: 0.013 seconds

	VACCINATION_ID	VACCINATION_NAM	VACCINATION_DATE	VACCINATION_DOSA
1	1	Rabies	1/15/2023, 12:00:00 A	1st dose
2	2	Canine Distemper	2/10/2023, 12:00:00 A	Booster
3	3	Feline Calicivirus	3/5/2023, 12:00:00 AM	Initial
4	4	Canine Parvovirus	4/20/2023, 12:00:00 A	Booster
5	5	Feline Panleukopenia	5/12/2023, 12:00:00 A	Initial
6	6	Canine Hepatitis	6/8/2023, 12:00:00 AM	Booster
7	7	Feline Leukemia	7/25/2023, 12:00:00 A	Initial
8	8	Canine Bordetella	8/18/2023, 12:00:00 A	Booster
9	9	Canine Influenza	9/3/2023, 12:00:00 AM	Initial
10	10	Feline Rabies	10/30/2023, 12:00:00	Booster

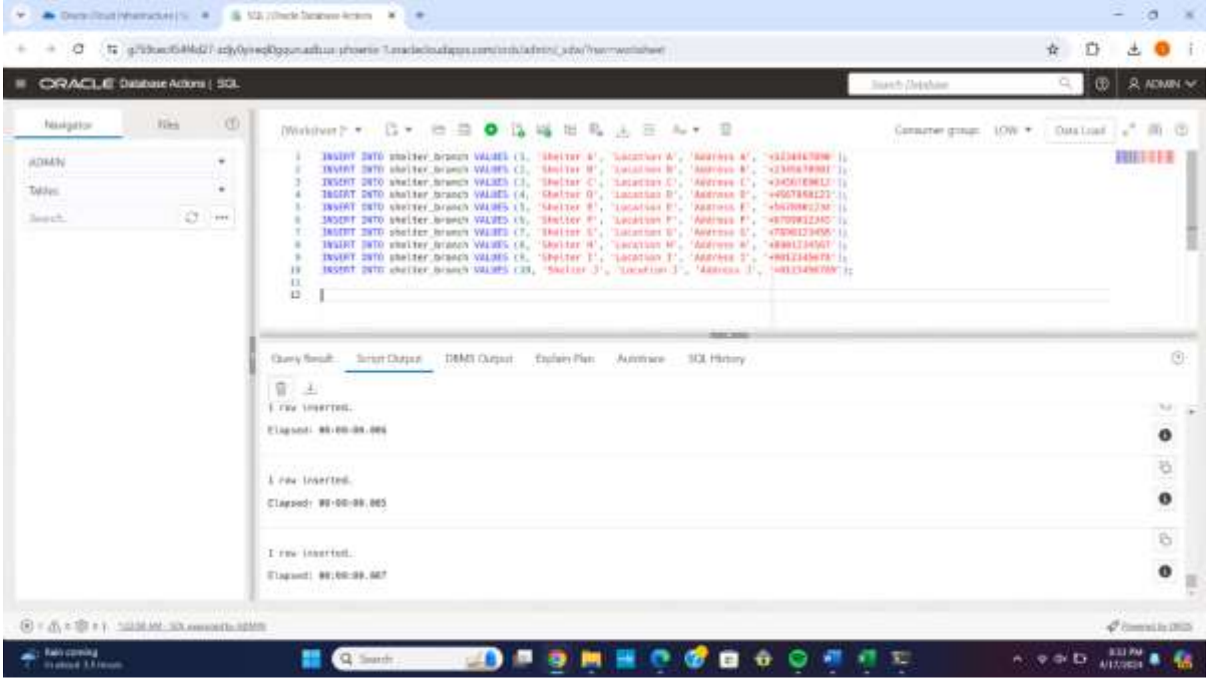
Creating a table named shelter_branch



```
CREATE TABLE shelter_branch (
    shelter_id INT PRIMARY KEY,
    shelter_name VARCHAR2(50) NOT NULL,
    shelter_loc VARCHAR2(35) NOT NULL,
    address VARCHAR2(100) NOT NULL,
    shelter_contact VARCHAR2(12) NOT NULL CHECK (shelter_contact NOT LIKE '% %')
);
```

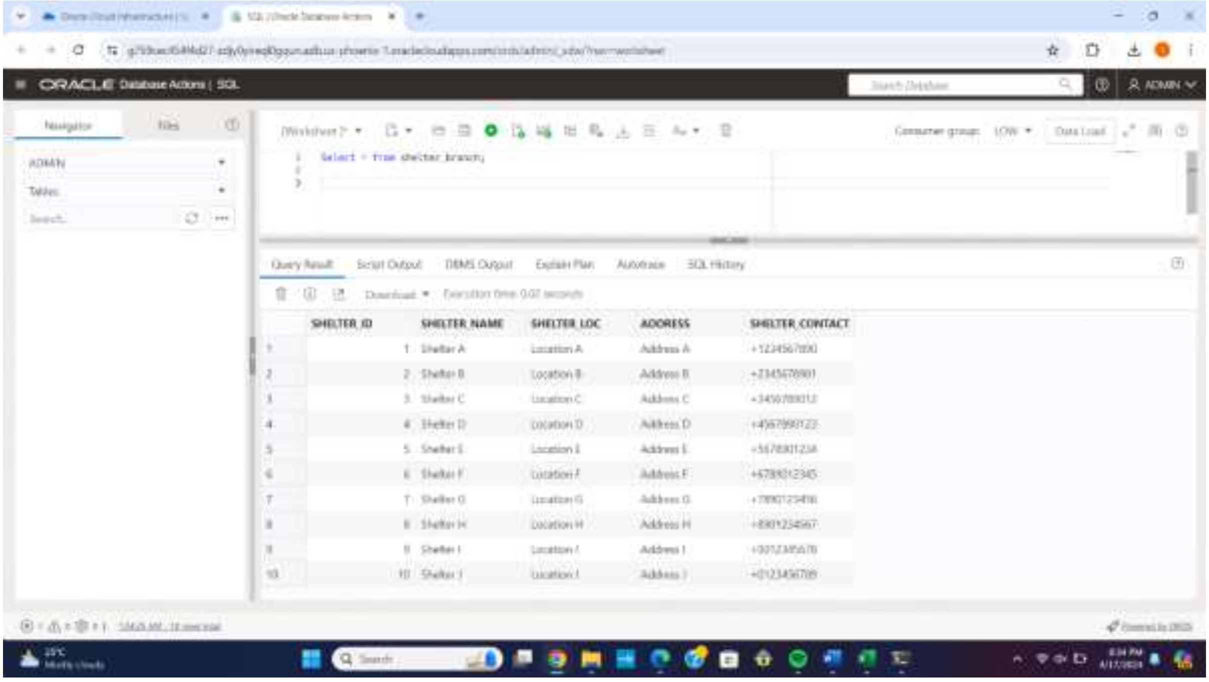
```
1 row inserted.

Elapsed: 00:00:00.003


1 row inserted.

Elapsed: 00:00:00.003


Table SHELTER_BRANCH created.

Elapsed: 00:00:00.025
```

Inserting values to shelter_branch



```
INSERT INTO shelter_branch VALUES (1, 'Shelter A', 'Location A', 'Address A', '+1234567890');
INSERT INTO shelter_branch VALUES (2, 'Shelter B', 'Location B', 'Address B', '+2345678901');
INSERT INTO shelter_branch VALUES (3, 'Shelter C', 'Location C', 'Address C', '+3456789012');
INSERT INTO shelter_branch VALUES (4, 'Shelter D', 'Location D', 'Address D', '+4567890123');
INSERT INTO shelter_branch VALUES (5, 'Shelter E', 'Location E', 'Address E', '+5678901234');
INSERT INTO shelter_branch VALUES (6, 'Shelter F', 'Location F', 'Address F', '+6789012345');
INSERT INTO shelter_branch VALUES (7, 'Shelter G', 'Location G', 'Address G', '+7890123456');
INSERT INTO shelter_branch VALUES (8, 'Shelter H', 'Location H', 'Address H', '+8901234567');
INSERT INTO shelter_branch VALUES (9, 'Shelter I', 'Location I', 'Address I', '+9012345678');
INSERT INTO shelter_branch VALUES (10, 'Shelter J', 'Location J', 'Address J', '+0123456789');
```

```
1 row inserted.

Elapsed: 00:00:00.006


1 row inserted.

Elapsed: 00:00:00.005


1 row inserted.

Elapsed: 00:00:00.007
```

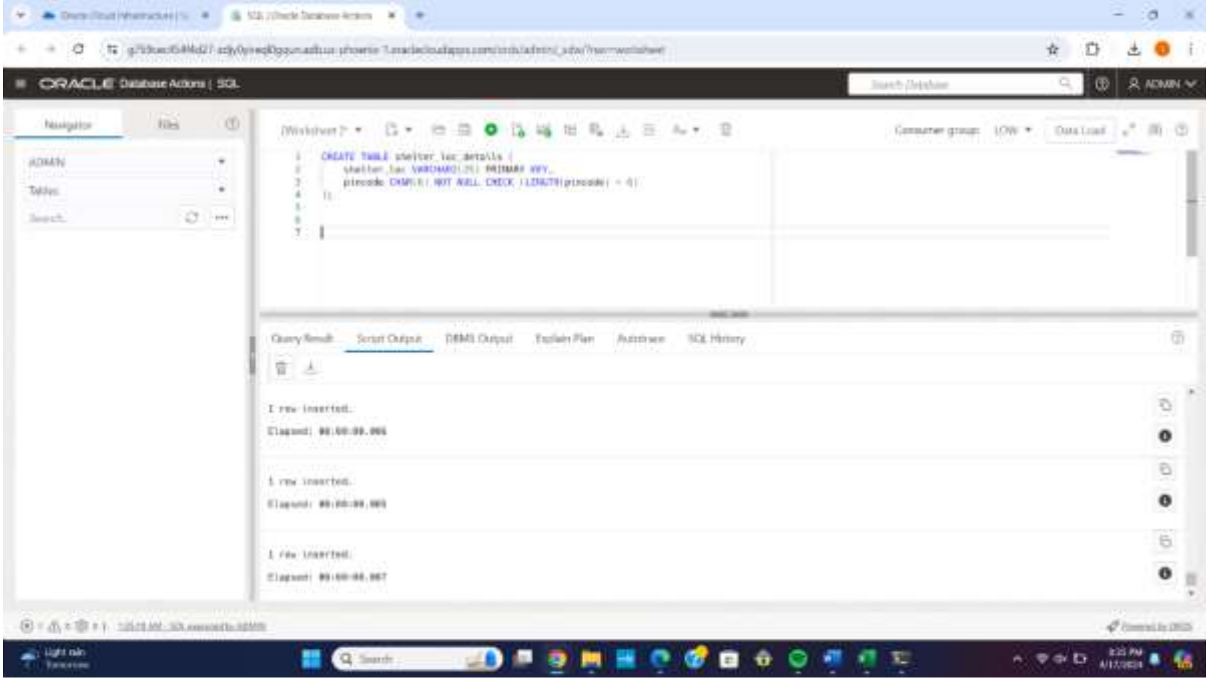


```
Select * from shelter_branch;
```

Execution time 0.07 seconds

	SHELTER_ID	SHELTER_NAME	SHELTER_LOC	ADDRESS	SHELTER_CONTACT
1	1	Shelter A	Location A	Address A	+1234567890
2	2	Shelter B	Location B	Address B	+2345678901
3	3	Shelter C	Location C	Address C	+3456789012
4	4	Shelter D	Location D	Address D	+4567890123
5	5	Shelter E	Location E	Address E	+5678901234
6	6	Shelter F	Location F	Address F	+6789012345
7	7	Shelter G	Location G	Address G	+7890123456
8	8	Shelter H	Location H	Address H	+8901234567
9	9	Shelter I	Location I	Address I	+9012345678
10	10	Shelter J	Location J	Address J	+0123456789

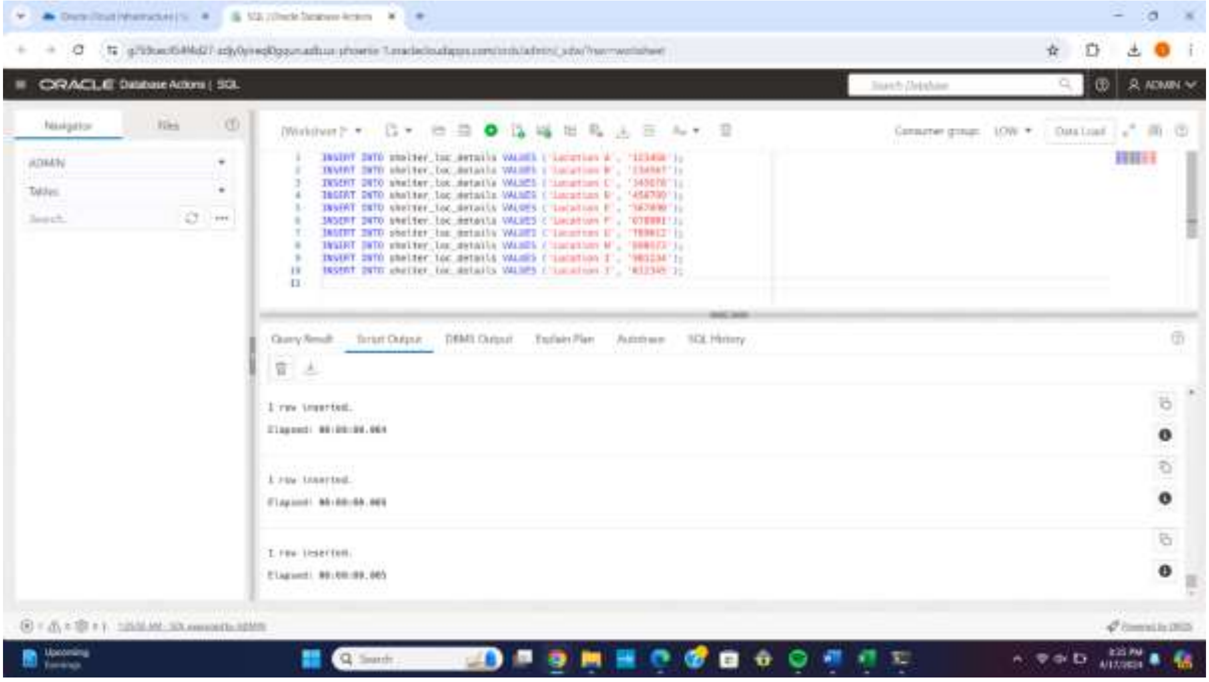
Creating a table named shelter_loc_details(New)



```
CREATE TABLE shelter_loc_details (
    shelter_loc VARCHAR2(25) PRIMARY KEY,
    pincode CHAR(6) NOT NULL CHECK (LENGTH(pincode) = 6)
);
```

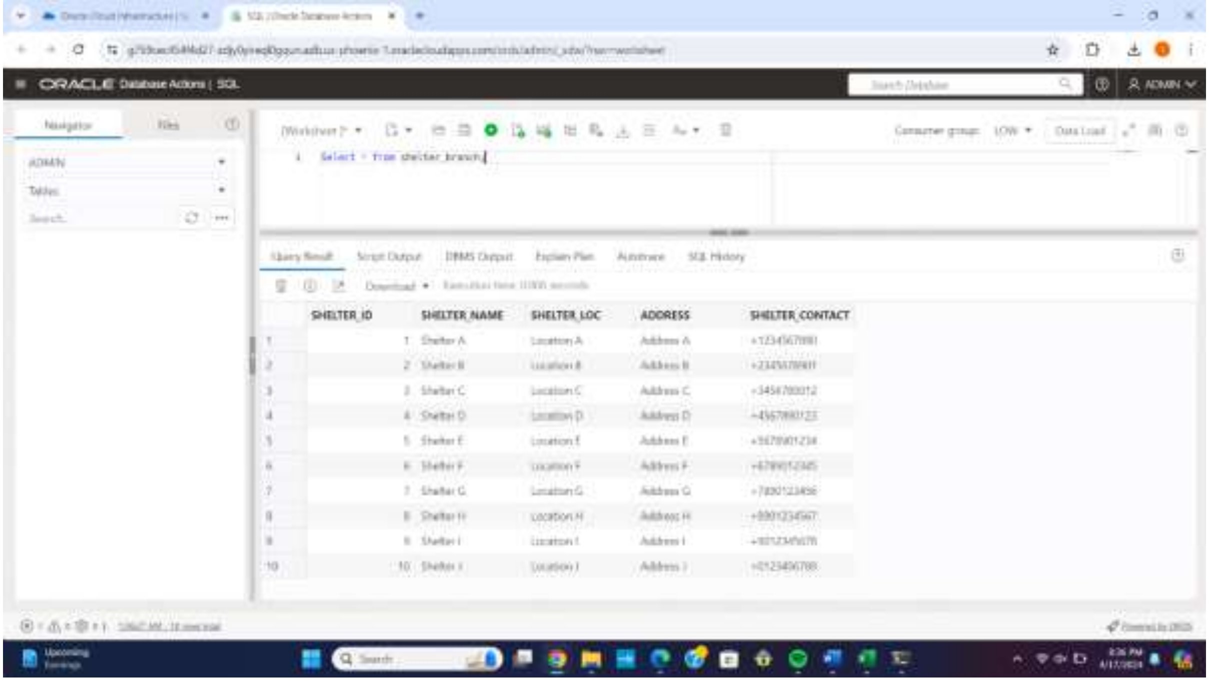
```
1 row inserted.

Elapsed: 00:00:00.006

1 row inserted.

Elapsed: 00:00:00.005

1 row inserted.

Elapsed: 00:00:00.007
```

Inserting values to shelter_loc_details(New)



```
INSERT INTO shelter_loc_details VALUES ('Location A', '123456');
INSERT INTO shelter_loc_details VALUES ('Location B', '234567');
INSERT INTO shelter_loc_details VALUES ('Location C', '345678');
INSERT INTO shelter_loc_details VALUES ('Location D', '456789');
INSERT INTO shelter_loc_details VALUES ('Location E', '567890');
INSERT INTO shelter_loc_details VALUES ('Location F', '678901');
INSERT INTO shelter_loc_details VALUES ('Location G', '789012');
INSERT INTO shelter_loc_details VALUES ('Location H', '890123');
INSERT INTO shelter_loc_details VALUES ('Location I', '901234');
INSERT INTO shelter_loc_details VALUES ('Location J', '012345');
```

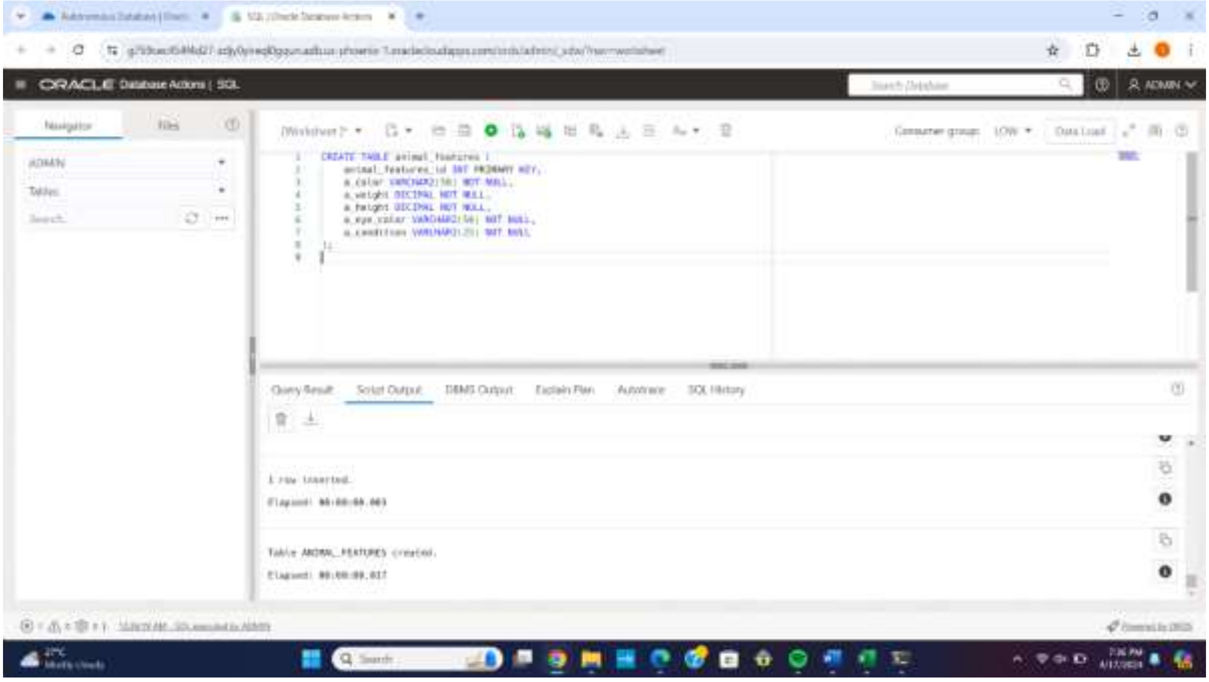
```
1 row inserted.

Elapsed: 00:00:00.004

1 row inserted.

Elapsed: 00:00:00.005

1 row inserted.

Elapsed: 00:00:00.005
```



```
Select * from shelter_branch;
```

	SHELTER_ID	SHELTER_NAME	SHELTER_LOC	ADDRESS	SHELTER_CONTACT
1	1	Shelter A	Location A	Address A	+1234567890
2	2	Shelter B	Location B	Address B	+2345678901
3	3	Shelter C	Location C	Address C	+3456789012
4	4	Shelter D	Location D	Address D	+4567890123
5	5	Shelter E	Location E	Address E	+5678901234
6	6	Shelter F	Location F	Address F	+6789012345
7	7	Shelter G	Location G	Address G	+7890123456
8	8	Shelter H	Location H	Address H	+8901234567
9	9	Shelter I	Location I	Address I	+9012345678
10	10	Shelter J	Location J	Address J	+0123456789

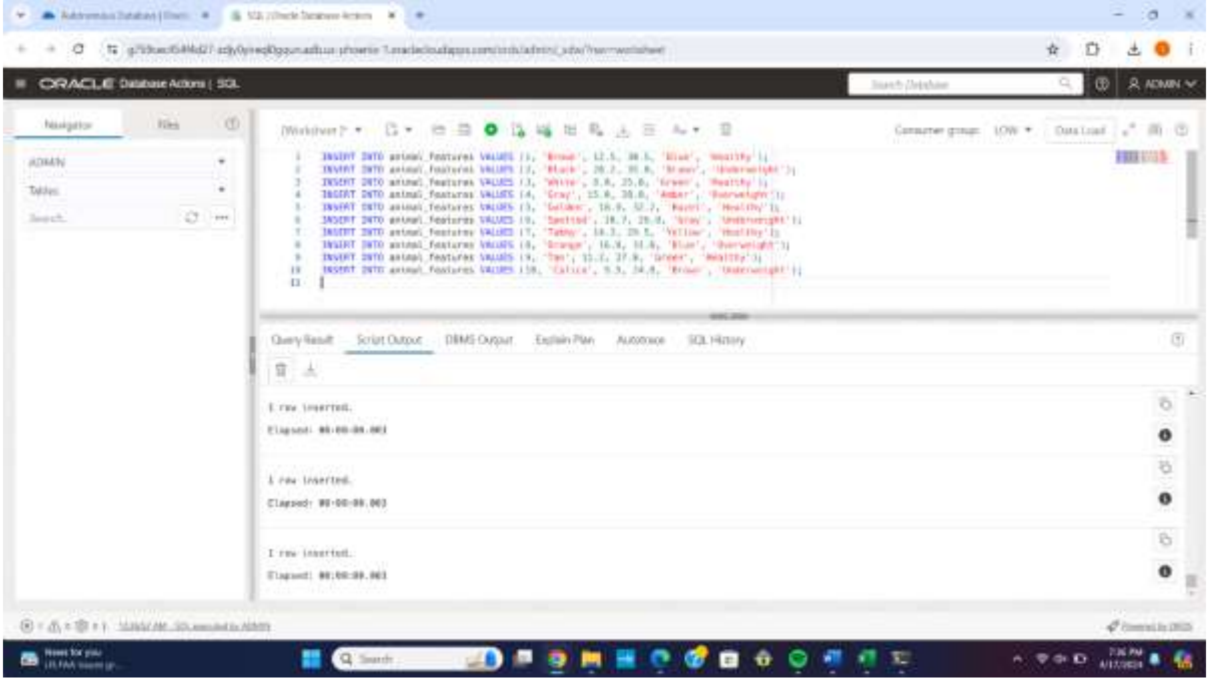
Creating a table named animal_features



```
CREATE TABLE animal_features (
    animal_features_id INT PRIMARY KEY,
    a_color VARCHAR2(50) NOT NULL,
    a_weight DECIMAL NOT NULL,
    a_height DECIMAL NOT NULL,
    a_eye_color VARCHAR2(50) NOT NULL,
    a_condition VARCHAR2(25) NOT NULL
);
```

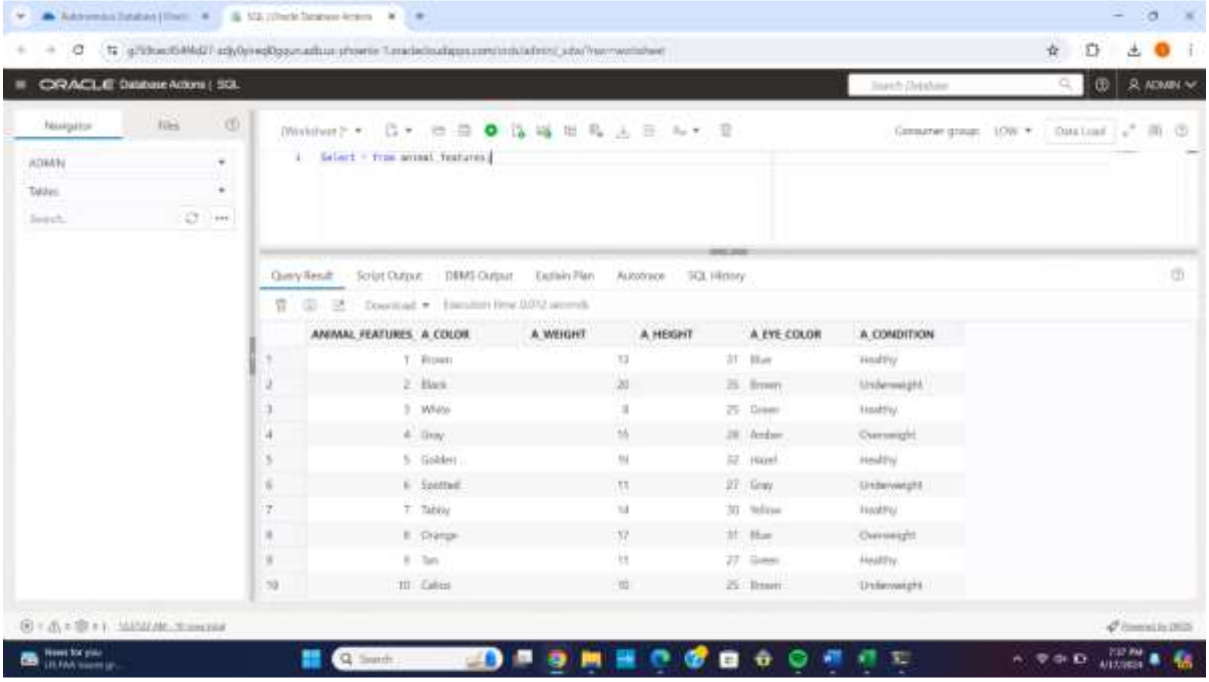
```
1 row inserted.

Elapsed: 00:00:00.003

Table ANIMAL_FEATURES created.

Elapsed: 00:00:00.017
```

Inserting values to animal_features



```
INSERT INTO animal_features VALUES (1, 'Brown', 12.5, 30.5, 'Blue', 'Healthy');
INSERT INTO animal_features VALUES (2, 'Black', 20.2, 35.0, 'Brown', 'Underweight');
INSERT INTO animal_features VALUES (3, 'White', 8.0, 25.0, 'Green', 'Healthy');
INSERT INTO animal_features VALUES (4, 'Gray', 15.0, 30.0, 'Amber', 'Overweight');
INSERT INTO animal_features VALUES (5, 'Golden', 18.9, 32.2, 'Hazel', 'Healthy');
INSERT INTO animal_features VALUES (6, 'Spotted', 10.7, 28.0, 'Gray', 'Underweight');
INSERT INTO animal_features VALUES (7, 'Tabby', 14.3, 29.5, 'Yellow', 'Healthy');
INSERT INTO animal_features VALUES (8, 'Orange', 16.8, 31.6, 'Blue', 'Overweight');
INSERT INTO animal_features VALUES (9, 'Tan', 11.2, 27.0, 'Green', 'Healthy');
INSERT INTO animal_features VALUES (10, 'Calico', 9.5, 24.0, 'Brown', 'Underweight');
```

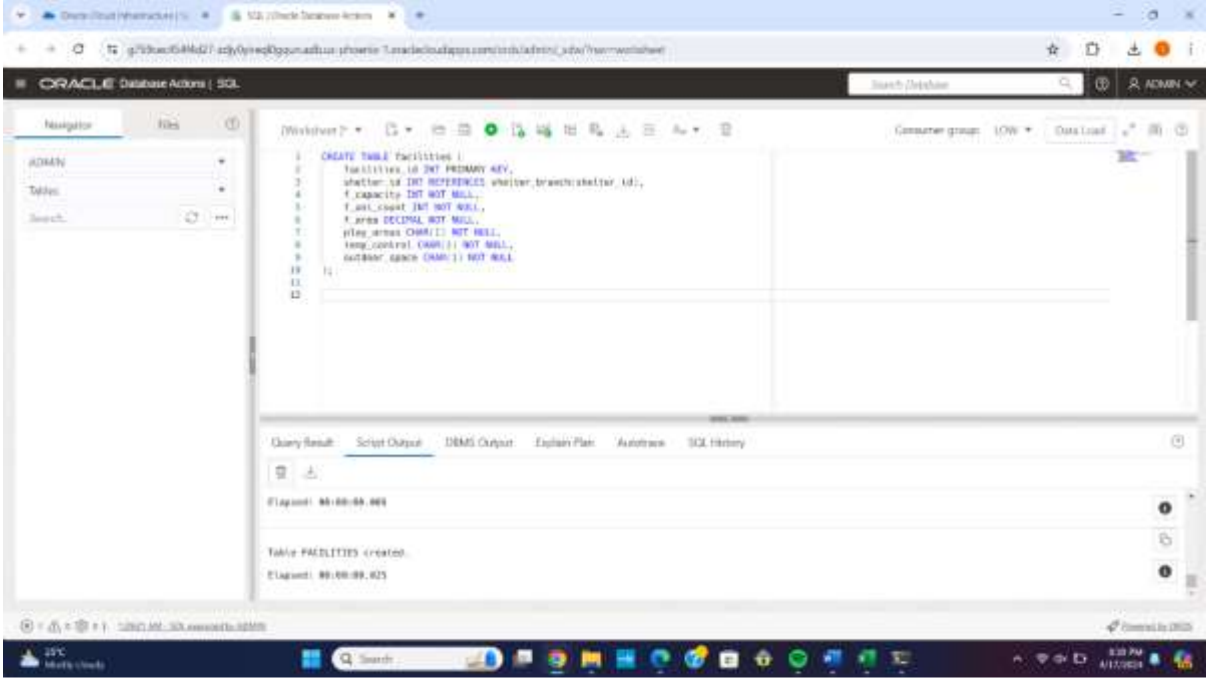
```
1 row inserted.

Elapsed: 00:00:00.003

1 row inserted.

Elapsed: 00:00:00.003

1 row inserted.

Elapsed: 00:00:00.003
```



```sql
Select * from animal_features;
```

	ANIMAL_FEATURES	A_COLOR	A_WEIGHT	A_HEIGHT	A_EYE_COLOR	A_CONDITION
1	1	Brown	13	31	Blue	Healthy
2	2	Black	20	35	Brown	Underweight
3	3	White	8	25	Green	Healthy
4	4	Gray	15	28	Amber	Overweight
5	5	Golden	19	32	Hazel	Healthy
6	6	Spotted	11	27	Gray	Underweight
7	7	Tabby	14	30	Yellow	Healthy
8	8	Orange	17	31	Blue	Overweight
9	9	Tan	11	27	Green	Healthy
10	10	Calico	10	25	Brown	Underweight

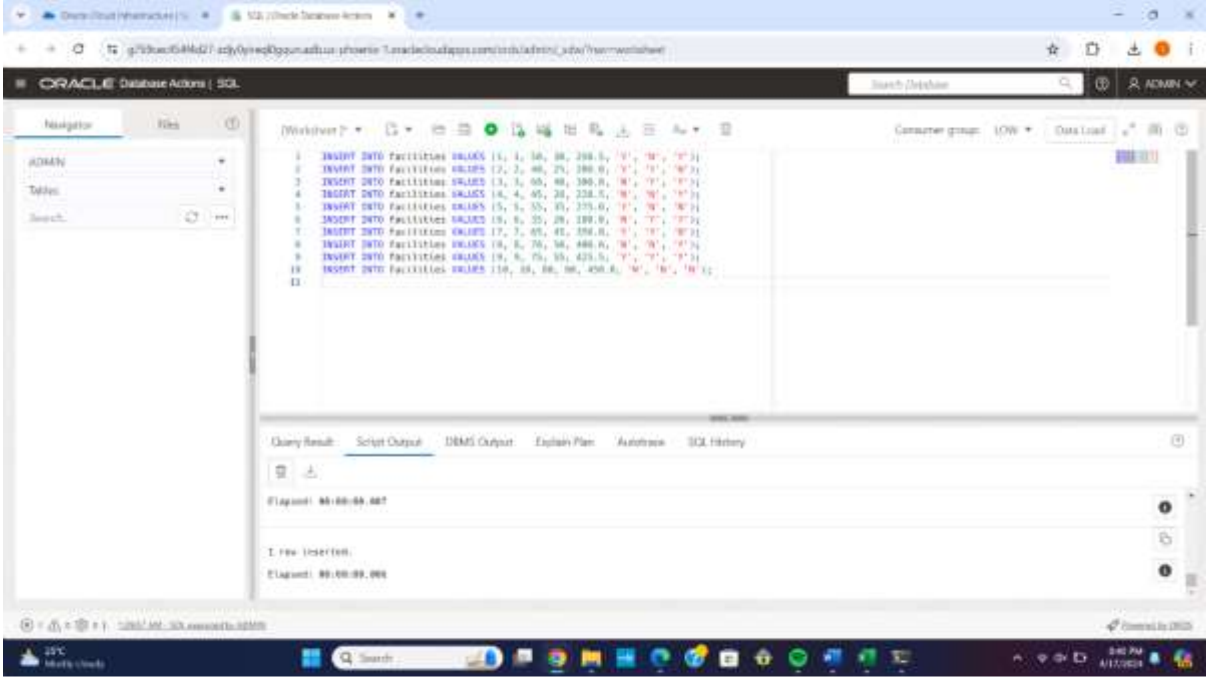
Creating a table named facilities



```sql
CREATE TABLE facilities (
    facilities_id INT PRIMARY KEY,
    shelter_id INT REFERENCES shelter_branch(shelter_id),
    f_capacity INT NOT NULL,
    f_ani_count INT NOT NULL,
    f_area DECIMAL NOT NULL,
    play_areas CHAR(1) NOT NULL,
    temp_control CHAR(1) NOT NULL,
    outdoor_space CHAR(1) NOT NULL
);
```

```
Elapsed: 00:00:00.003

Table FACILITIES created.

Elapsed: 00:00:00.025
```

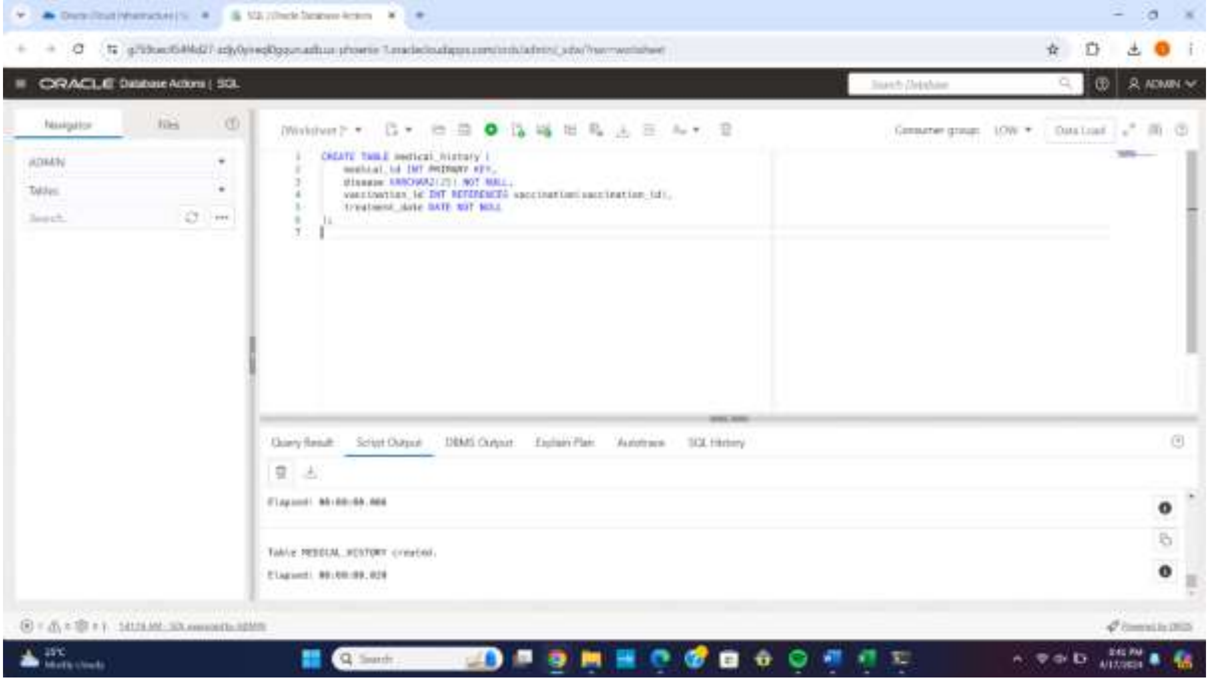
Inserting values to facilities



```
INSERT INTO facilities VALUES (1, 1, 50, 30, 250.5, 'Y', 'N', 'Y');
INSERT INTO facilities VALUES (2, 2, 40, 25, 200.0, 'Y', 'Y', 'N');
INSERT INTO facilities VALUES (3, 3, 60, 40, 300.0, 'N', 'Y', 'Y');
INSERT INTO facilities VALUES (4, 4, 45, 20, 220.5, 'N', 'N', 'Y');
INSERT INTO facilities VALUES (5, 5, 55, 35, 275.0, 'Y', 'N', 'N');
INSERT INTO facilities VALUES (6, 6, 35, 20, 180.0, 'N', 'Y', 'Y');
INSERT INTO facilities VALUES (7, 7, 65, 45, 350.0, 'Y', 'Y', 'N');
INSERT INTO facilities VALUES (8, 8, 70, 50, 400.0, 'N', 'N', 'Y');
INSERT INTO facilities VALUES (9, 9, 75, 55, 425.5, 'Y', 'Y', 'Y');
INSERT INTO facilities VALUES (10, 10, 80, 60, 450.0, 'N', 'N', 'N');
```

```
Elapsed: 00:00:00.007

1 row inserted.

Elapsed: 00:00:00.006
```

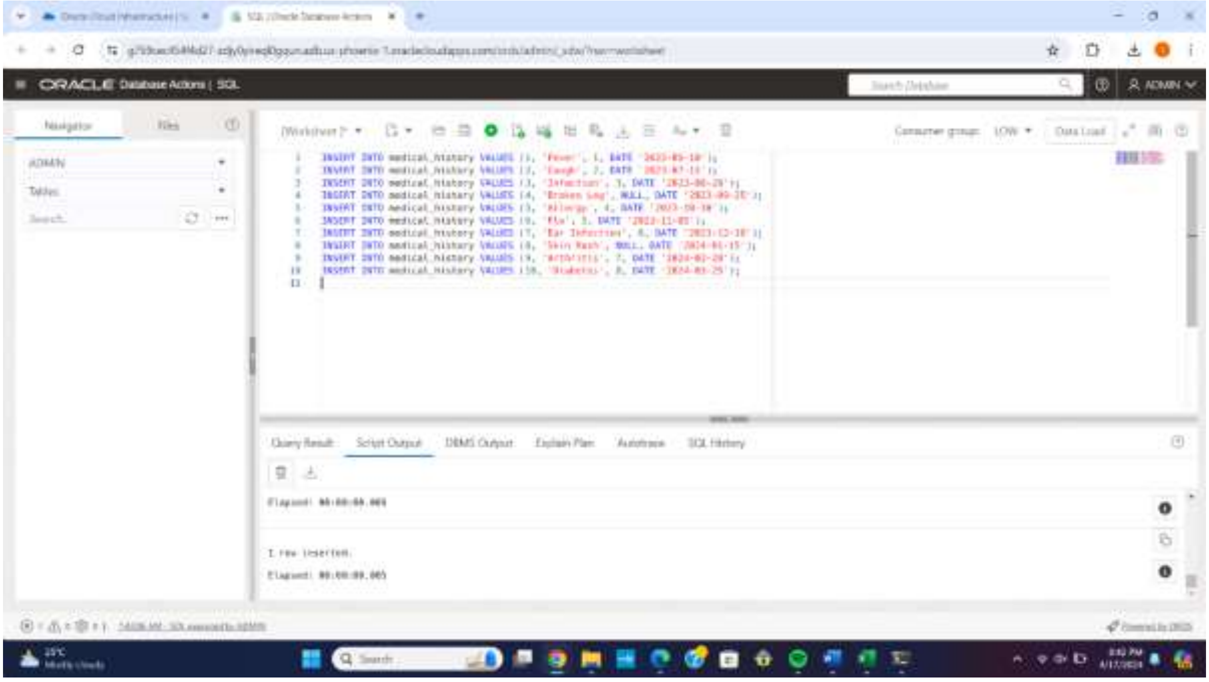
Creating a table named medical_history



```
CREATE TABLE medical_history (
    medical_id INT PRIMARY KEY,
    disease VARCHAR2(25) NOT NULL,
    vaccination_id INT REFERENCES vaccination(vaccination_id),
    treatment_date DATE NOT NULL
);
```

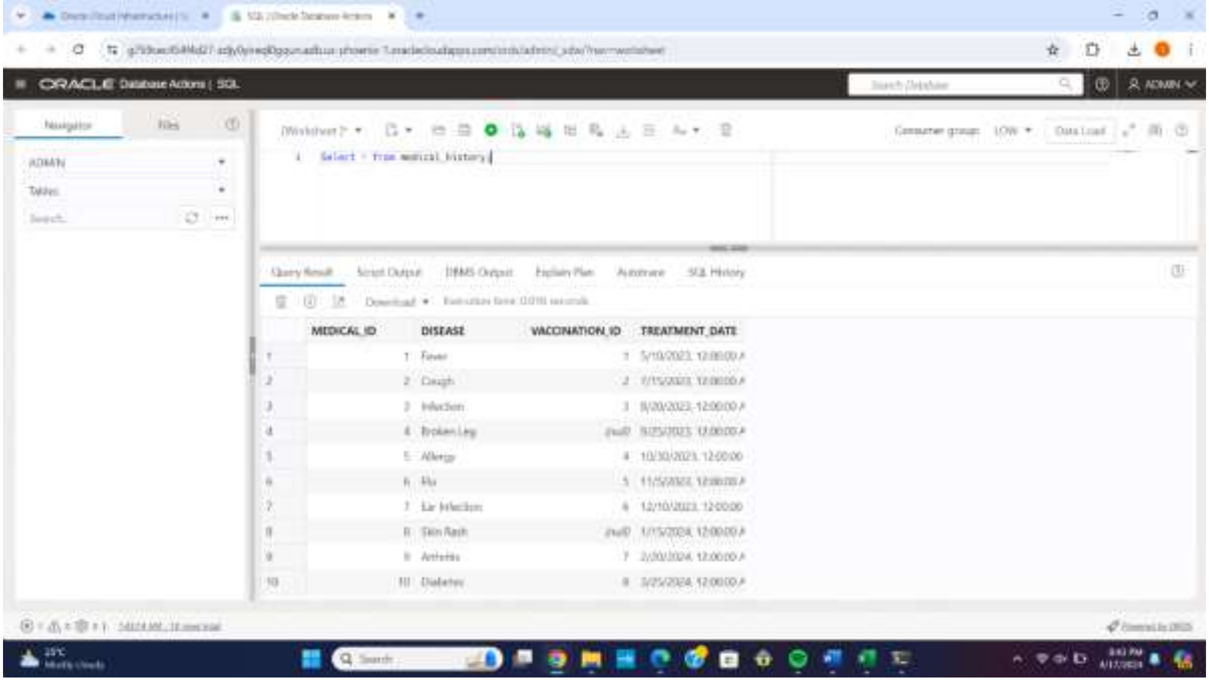
```
Elapsed: 00:00:00.004

Table MEDICAL_HISTORY created.

Elapsed: 00:00:00.028
```

Inserting values to medical_history



```
INSERT INTO medical_history VALUES (1, 'Fever', 1, DATE '2023-05-10');
INSERT INTO medical_history VALUES (2, 'Cough', 2, DATE '2023-07-15');
INSERT INTO medical_history VALUES (3, 'Infection', 3, DATE '2023-08-20');
INSERT INTO medical_history VALUES (4, 'Broken Leg', NULL, DATE '2023-09-25');
INSERT INTO medical_history VALUES (5, 'Allergy', 4, DATE '2023-10-30');
INSERT INTO medical_history VALUES (6, 'Flu', 5, DATE '2023-11-05');
INSERT INTO medical_history VALUES (7, 'Ear Infection', 6, DATE '2023-12-10');
INSERT INTO medical_history VALUES (8, 'Skin Rash', NULL, DATE '2024-01-15');
INSERT INTO medical_history VALUES (9, 'Arthritis', 7, DATE '2024-02-20');
INSERT INTO medical_history VALUES (10, 'Diabetes', 8, DATE '2024-03-25');
```

```
Elapsed: 00:00:00.005


1 row inserted.

Elapsed: 00:00:00.005
```

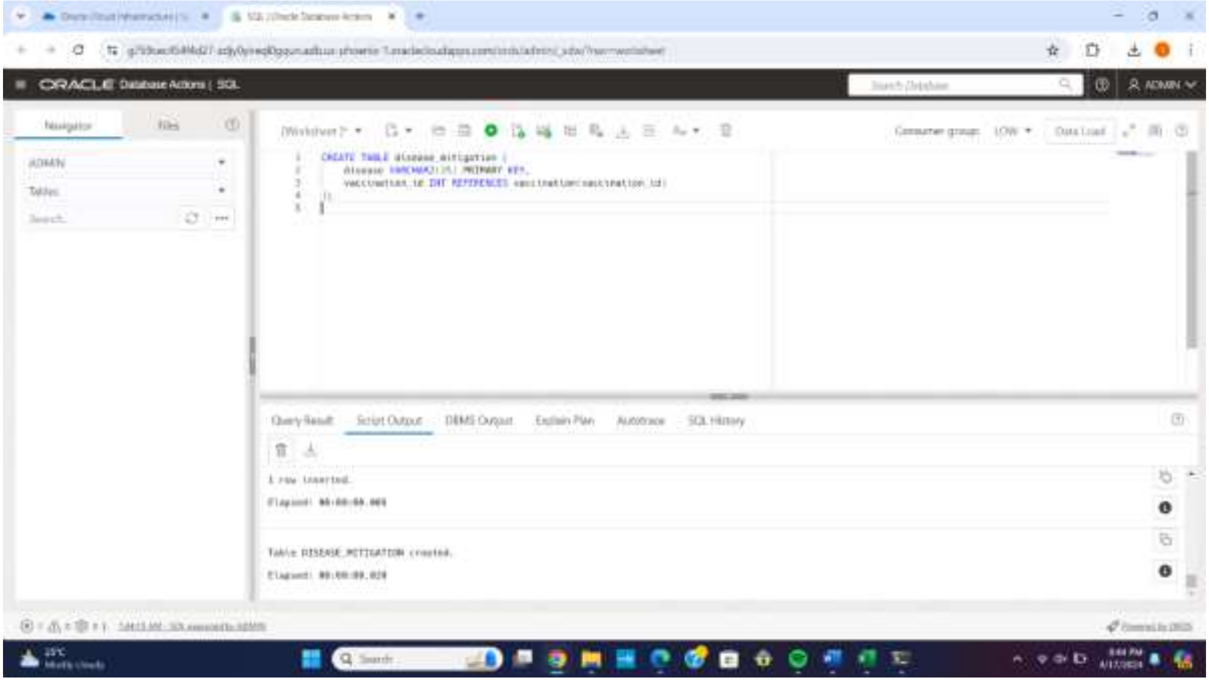


```
Select * from medical_history
```

Execution time: 0.018 seconds

	MEDICAL_ID	DISEASE	VACCINATION_ID	TREATMENT_DATE
1	1	Fever	1	5/10/2023, 12:00:00 A
2	2	Cough	2	7/15/2023, 12:00:00 A
3	3	Infection	3	8/20/2023, 12:00:00 A
4	4	Broken Leg	(null)	9/25/2023, 12:00:00 A
5	5	Allergy	4	10/30/2023, 12:00:00
6	6	Flu	5	11/5/2023, 12:00:00 A
7	7	Ear Infection	6	12/10/2023, 12:00:00
8	8	Skin Rash	(null)	1/15/2024, 12:00:00 A
9	9	Arthritis	7	2/20/2024, 12:00:00 A
10	10	Diabetes	8	3/25/2024, 12:00:00 A

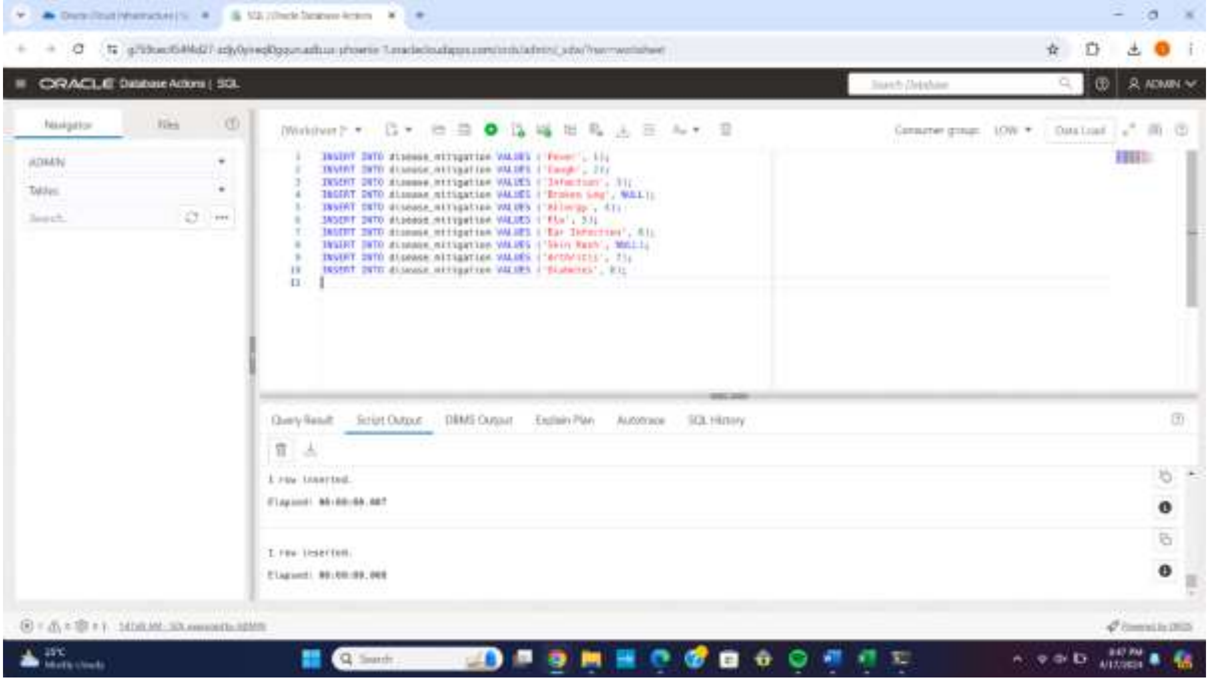
Creating a table named disease_mitigation (New)



```sql
CREATE TABLE disease_mitigation (
    disease VARCHAR2(25) PRIMARY KEY,
    vaccination_id INT REFERENCES vaccination(vaccination_id)
);
```

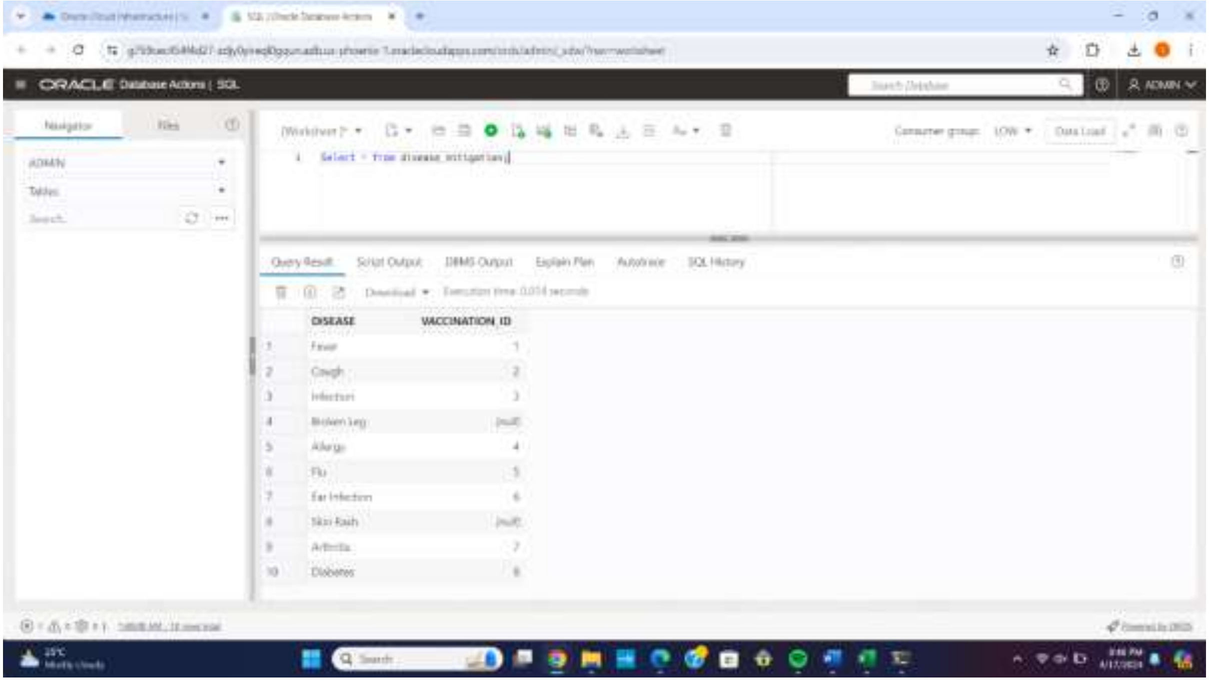
```
1 row inserted.

Elapsed: 00:00:00.003

Table DISEASE_MITIGATION created.

Elapsed: 00:00:00.020
```

Inserting values to disease_mitigation (New)



```sql
INSERT INTO disease_mitigation VALUES ('Fever', 1);
INSERT INTO disease_mitigation VALUES ('Cough', 2);
INSERT INTO disease_mitigation VALUES ('Infection', 3);
INSERT INTO disease_mitigation VALUES ('Broken Leg', NULL);
INSERT INTO disease_mitigation VALUES ('Allergy', 4);
INSERT INTO disease_mitigation VALUES ('Flu', 5);
INSERT INTO disease_mitigation VALUES ('Ear Infection', 6);
INSERT INTO disease_mitigation VALUES ('Skin Rash', NULL);
INSERT INTO disease_mitigation VALUES ('Arthritis', 7);
INSERT INTO disease_mitigation VALUES ('Diabetes', 8);
```

```
1 row inserted.

Elapsed: 00:00:00.007

1 row inserted.

Elapsed: 00:00:00.008
```

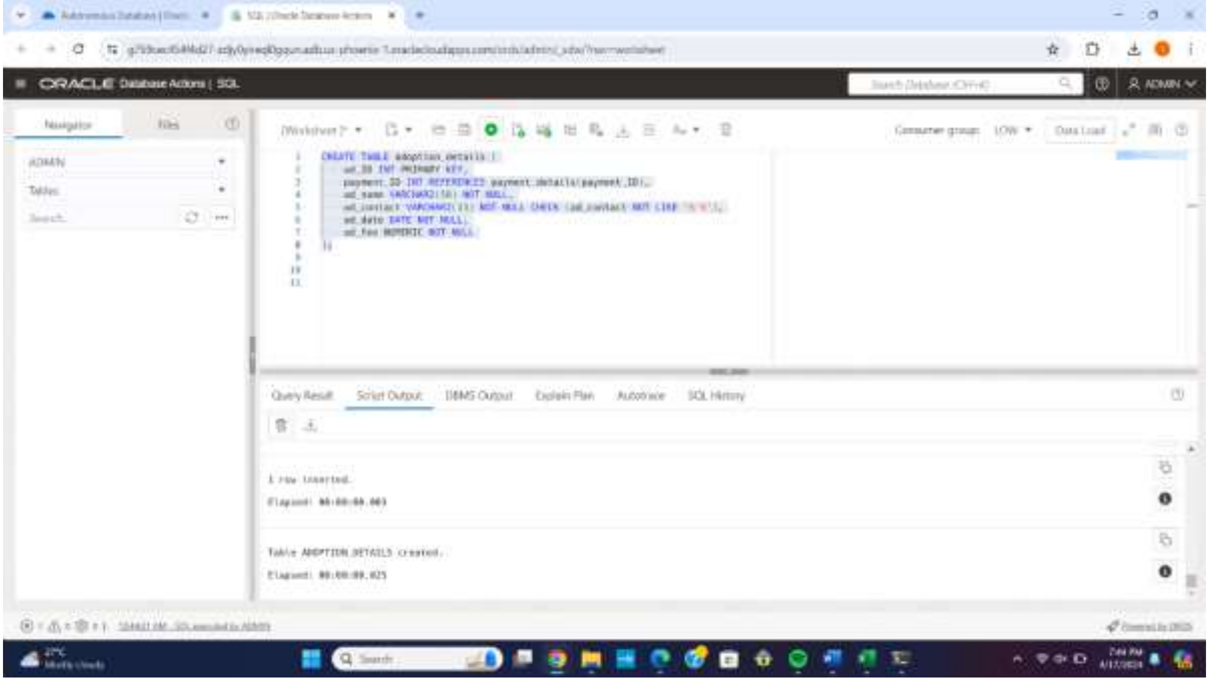


```
Select * from disease_mitigation;
```

Execution time 0.014 seconds

	DISEASE	VACCINATION_ID
1	Fever	1
2	Cough	2
3	Infection	3
4	Broken Leg	(null)
5	Allergy	4
6	Flu	5
7	Ear Infection	6
8	Skin Rash	(null)
9	Arthritis	7
10	Diabetes	8

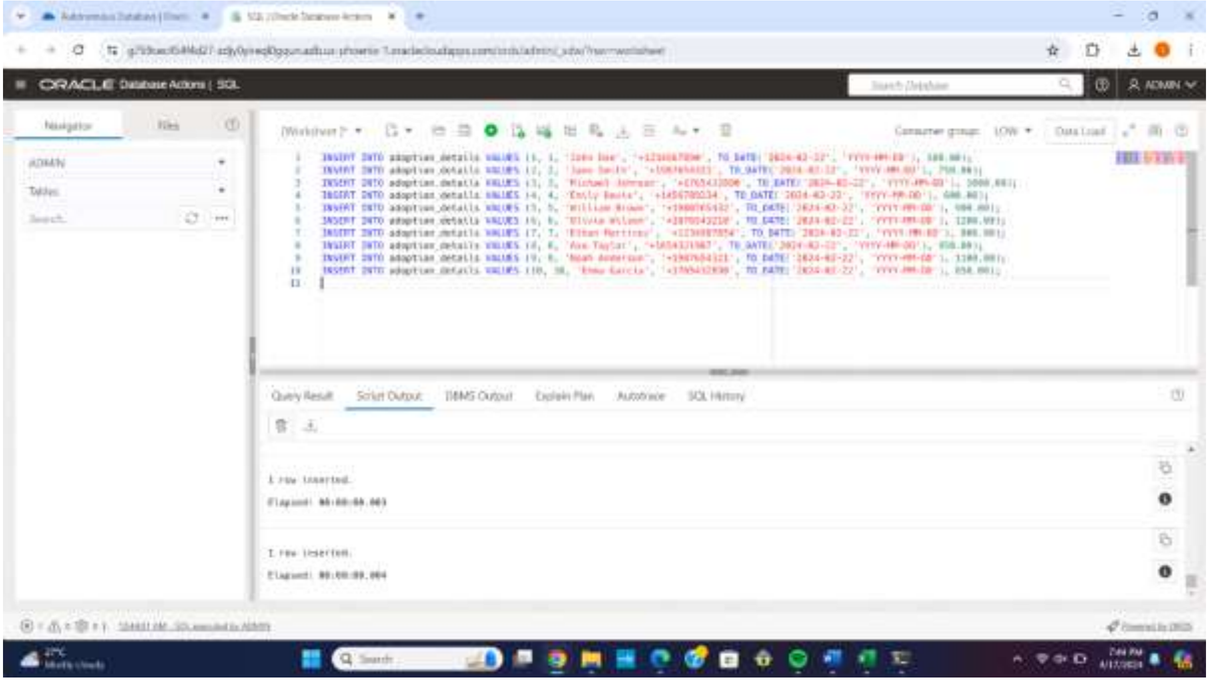
Creating a table named adoption_details



```
CREATE TABLE adoption_details (
    ad_ID INT PRIMARY KEY,
    payment_ID INT REFERENCES payment_details(payment_ID),
    ad_name VARCHAR2(50) NOT NULL,
    ad_contact VARCHAR2(11) NOT NULL CHECK (ad_contact NOT LIKE '%[illegible]%'),
    ad_date DATE NOT NULL,
    ad_fee NUMERIC NOT NULL
);
```

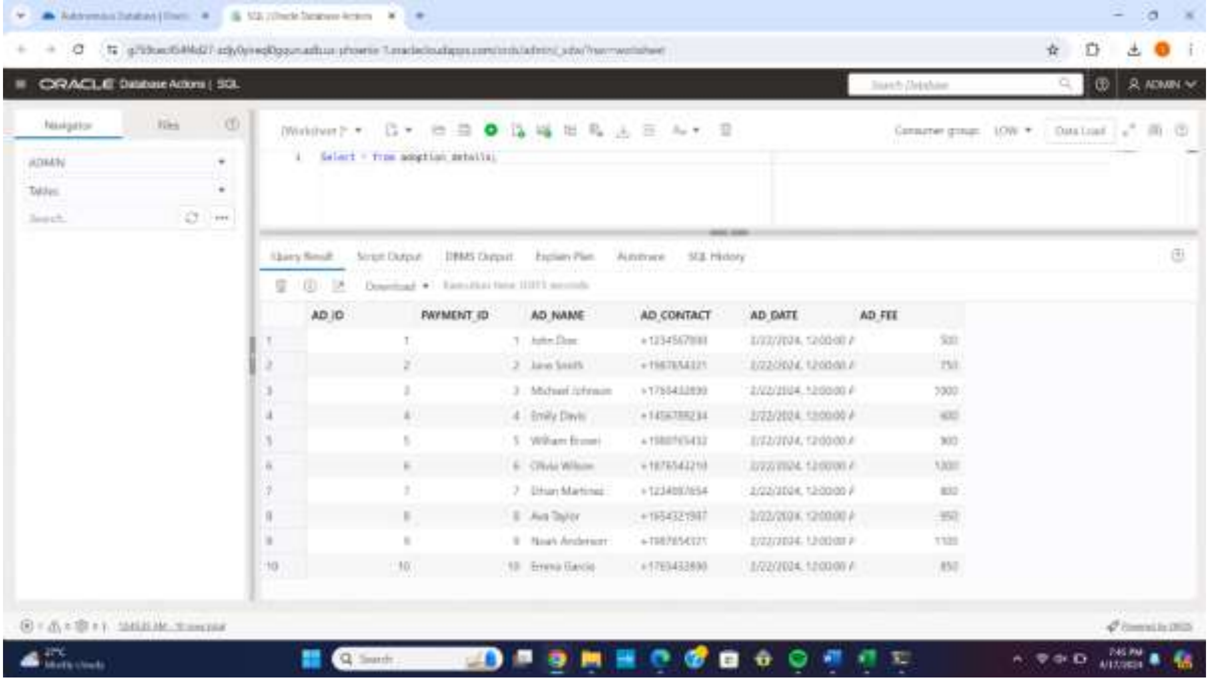
```
1 row inserted.

Elapsed: 00:00:00.003

Table ADOPTION_DETAILS created.

Elapsed: 00:00:00.025
```

Inserting values to adoption_details



```
INSERT INTO adoption_details VALUES (1, 1, 'John Doe', '+1234567890', TO_DATE('2024-02-22', 'YYYY-MM-DD'), 500.00);
INSERT INTO adoption_details VALUES (2, 2, 'Jane Smith', '+1987654321', TO_DATE('2024-02-22', 'YYYY-MM-DD'), 750.00);
INSERT INTO adoption_details VALUES (3, 3, 'Michael Johnson', '+1765432890', TO_DATE('2024-02-22', 'YYYY-MM-DD'), 1000.00);
INSERT INTO adoption_details VALUES (4, 4, 'Emily Davis', '+1456789234', TO_DATE('2024-02-22', 'YYYY-MM-DD'), 400.00);
INSERT INTO adoption_details VALUES (5, 5, 'William Brown', '+1980765432', TO_DATE('2024-02-22', 'YYYY-MM-DD'), 900.00);
INSERT INTO adoption_details VALUES (6, 6, 'Olivia Wilson', '+1876543210', TO_DATE('2024-02-22', 'YYYY-MM-DD'), 1200.00);
INSERT INTO adoption_details VALUES (7, 7, 'Ethan Martinez', '+1234987654', TO_DATE('2024-02-22', 'YYYY-MM-DD'), 800.00);
INSERT INTO adoption_details VALUES (8, 8, 'Ava Taylor', '+1654321987', TO_DATE('2024-02-22', 'YYYY-MM-DD'), 950.00);
INSERT INTO adoption_details VALUES (9, 9, 'Noah Anderson', '+1987654321', TO_DATE('2024-02-22', 'YYYY-MM-DD'), 1100.00);
INSERT INTO adoption_details VALUES (10, 10, 'Emma Garcia', '+1765432890', TO_DATE('2024-02-22', 'YYYY-MM-DD'), 850.00);
```

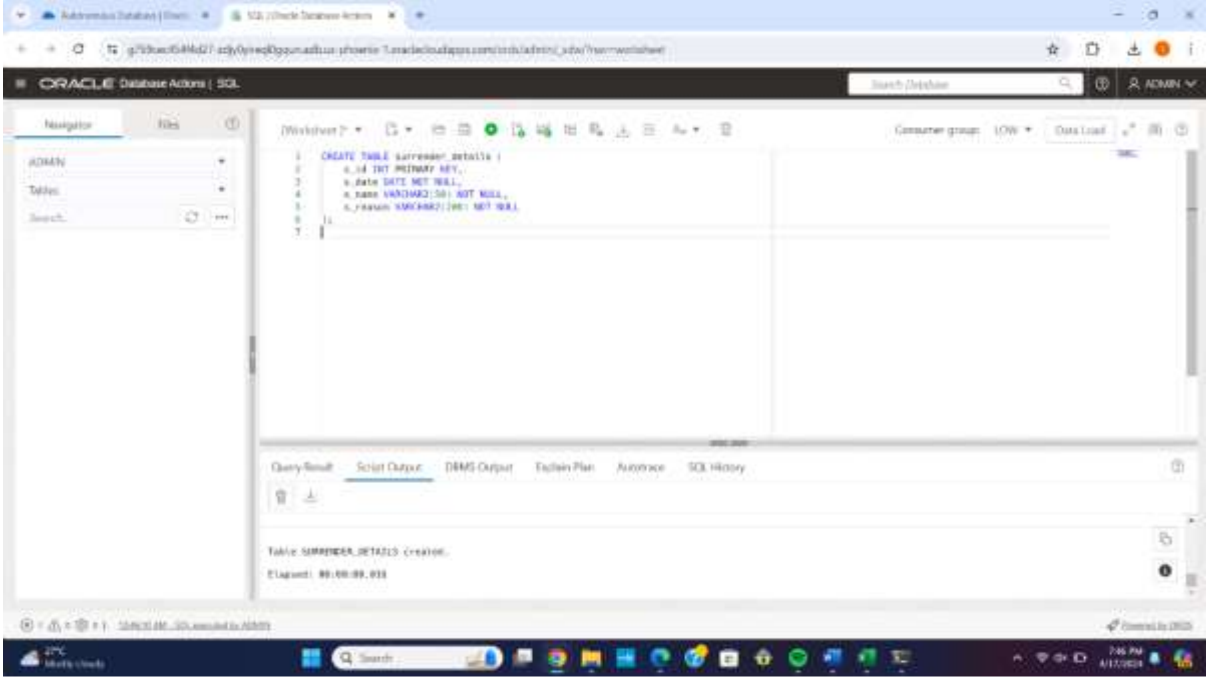
```
1 row inserted.

Elapsed: 00:00:00.003

1 row inserted.

Elapsed: 00:00:00.004
```



```
Select * from adoption_details;
```

	AD_ID	PAYMENT_ID	AD_NAME	AD_CONTACT	AD_DATE	AD_FEE
1	1	1	John Doe	+1234567890	2/22/2024, 12:00:00 A	500
2	2	2	Jane Smith	+1987654321	2/22/2024, 12:00:00 A	750
3	3	3	Michael Johnson	+1765432890	2/22/2024, 12:00:00 A	1000
4	4	4	Emily Davis	+1456789234	2/22/2024, 12:00:00 A	400
5	5	5	William Brown	+1980765432	2/22/2024, 12:00:00 A	900
6	6	6	Olivia Wilson	+1876543210	2/22/2024, 12:00:00 A	1200
7	7	7	Ethan Martinez	+1234987654	2/22/2024, 12:00:00 A	800
8	8	8	Ava Taylor	+1654321987	2/22/2024, 12:00:00 A	950
9	9	9	Noah Anderson	+1987654321	2/22/2024, 12:00:00 A	1100
10	10	10	Emma Garcia	+1765432890	2/22/2024, 12:00:00 A	850

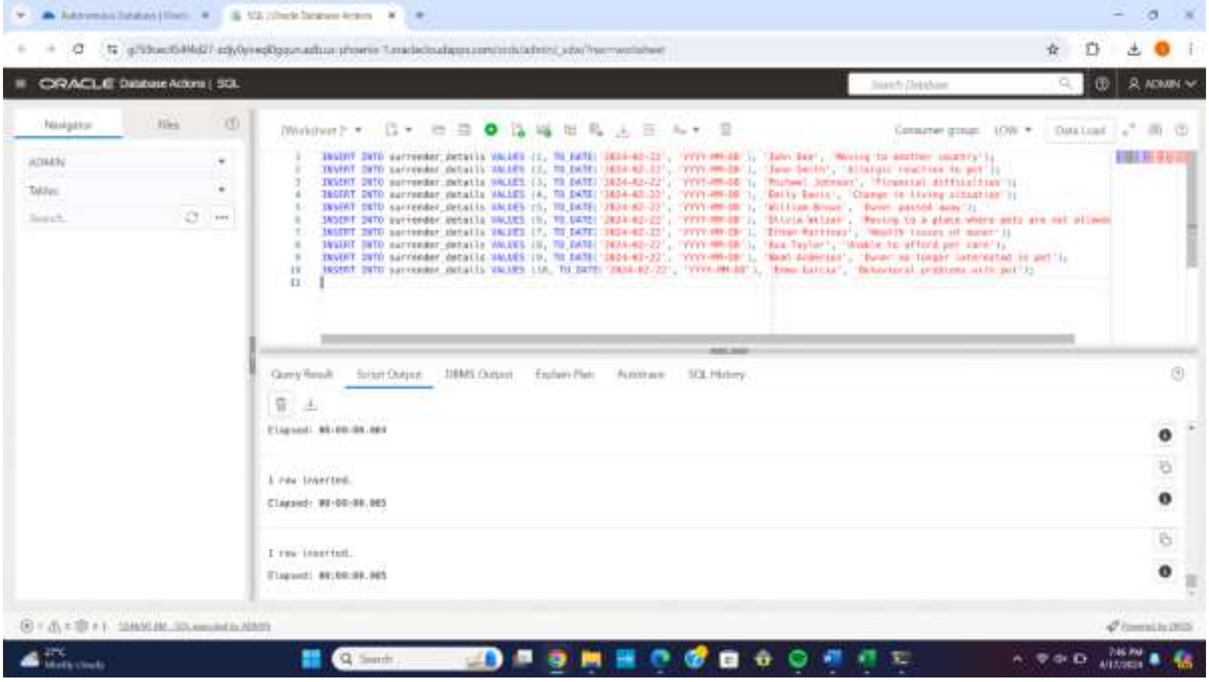
Creating a table named surrender_details



```sql
CREATE TABLE surrender_details (
    s_id INT PRIMARY KEY,
    s_date DATE NOT NULL,
    s_name VARCHAR2(50) NOT NULL,
    s_reason VARCHAR2(200) NOT NULL
);
```

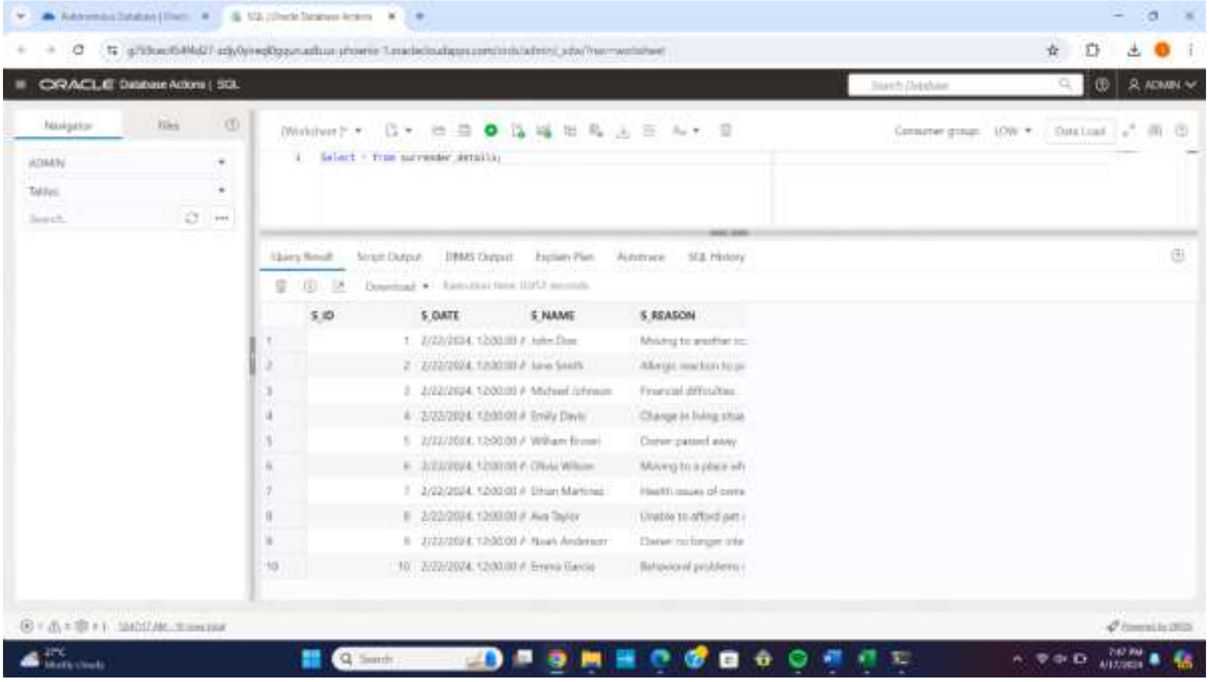
```
Table SURRENDER_DETAILS created.

Elapsed: 00:00:00.018
```

Inserting values to surrender_details



```sql
INSERT INTO surrender_details VALUES (1, TO_DATE('2024-02-22', 'YYYY-MM-DD'), 'John Doe', 'Moving to another country');
INSERT INTO surrender_details VALUES (2, TO_DATE('2024-02-22', 'YYYY-MM-DD'), 'Jane Smith', 'Allergic reaction to pet');
INSERT INTO surrender_details VALUES (3, TO_DATE('2024-02-22', 'YYYY-MM-DD'), 'Michael Johnson', 'Financial difficulties');
INSERT INTO surrender_details VALUES (4, TO_DATE('2024-02-22', 'YYYY-MM-DD'), 'Emily Davis', 'Change in living situation');
INSERT INTO surrender_details VALUES (5, TO_DATE('2024-02-22', 'YYYY-MM-DD'), 'William Brown', 'Owner passed away');
INSERT INTO surrender_details VALUES (6, TO_DATE('2024-02-22', 'YYYY-MM-DD'), 'Olivia Wilson', 'Moving to a place where pets are not allowed
INSERT INTO surrender_details VALUES (7, TO_DATE('2024-02-22', 'YYYY-MM-DD'), 'Ethan Martinez', 'Health issues of owner');
INSERT INTO surrender_details VALUES (8, TO_DATE('2024-02-22', 'YYYY-MM-DD'), 'Ava Taylor', 'Unable to afford pet care');
INSERT INTO surrender_details VALUES (9, TO_DATE('2024-02-22', 'YYYY-MM-DD'), 'Noah Anderson', 'Owner no longer interested in pet');
INSERT INTO surrender_details VALUES (10, TO_DATE('2024-02-22', 'YYYY-MM-DD'), 'Emma Garcia', 'Behavioral problems with pet');
```

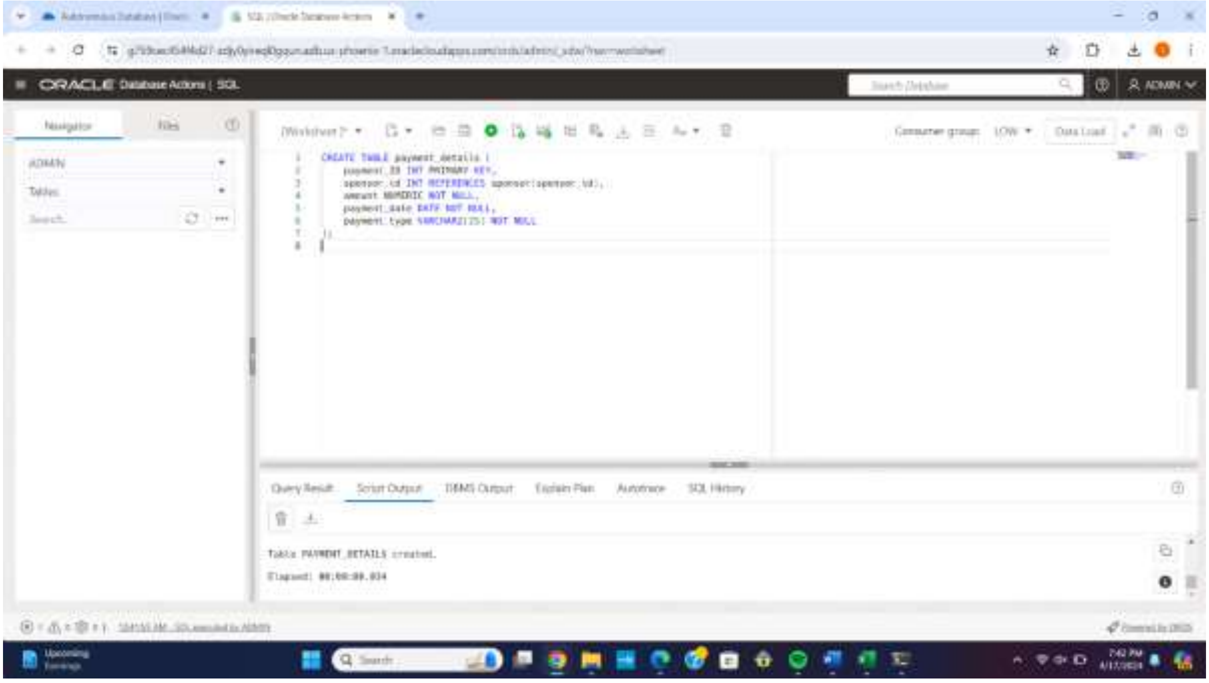
```
Elapsed: 00:00:00.004

1 row inserted.

Elapsed: 00:00:00.005

1 row inserted.

Elapsed: 00:00:00.005
```



```sql
Select * from surrender_details;
```

	S_ID	S_DATE	S_NAME	S_REASON
1	1	2/22/2024, 12:00:00 A	John Doe	Moving to another sc
2	2	2/22/2024, 12:00:00 A	Jane Smith	Allergic reaction to pe
3	3	2/22/2024, 12:00:00 A	Michael Johnson	Financial difficulties
4	4	2/22/2024, 12:00:00 A	Emily Davis	Change in living situa
5	5	2/22/2024, 12:00:00 A	William Brown	Owner passed away
6	6	2/22/2024, 12:00:00 A	Olivia Wilson	Moving to a place wh
7	7	2/22/2024, 12:00:00 A	Ethan Martinez	Health issues of owne
8	8	2/22/2024, 12:00:00 A	Ava Taylor	Unable to afford pet c
9	9	2/22/2024, 12:00:00 A	Noah Anderson	Owner no longer inte
10	10	2/22/2024, 12:00:00 A	Emma Garcia	Behavioral problems i

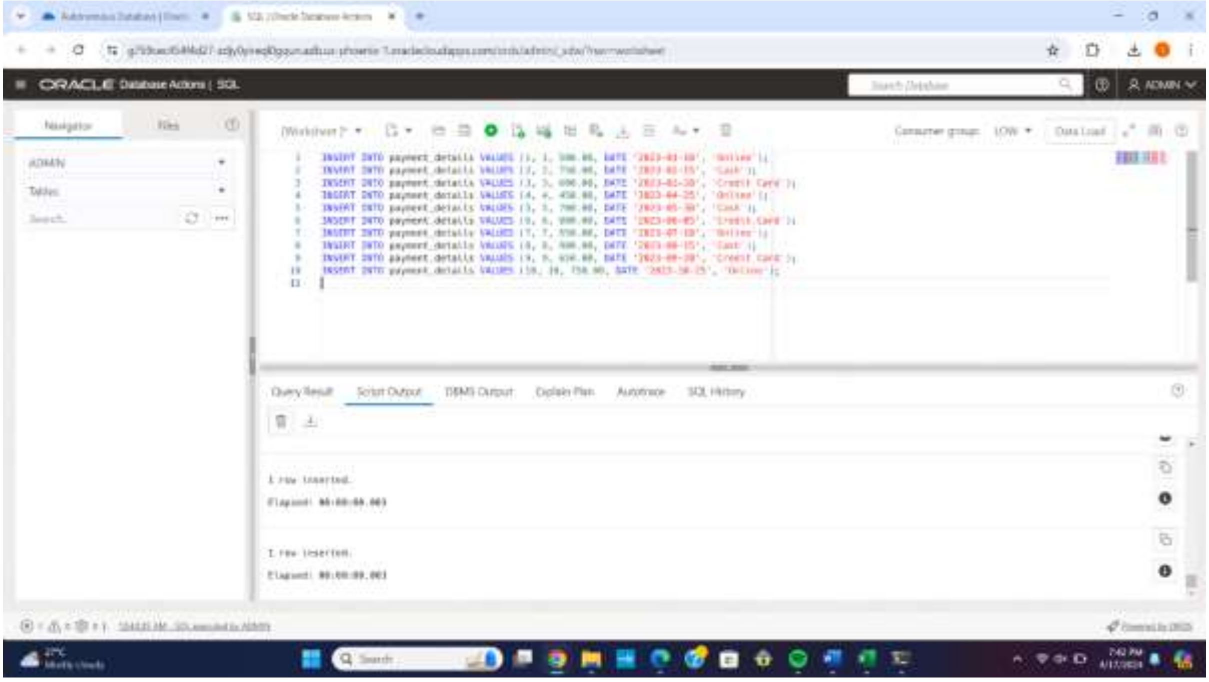
Creating a table named payment_details



```sql
CREATE TABLE payment_details (
    payment_ID INT PRIMARY KEY,
    sponsor_id INT REFERENCES sponsor(sponsor_id),
    amount NUMERIC NOT NULL,
    payment_date DATE NOT NULL,
    payment_type VARCHAR2(25) NOT NULL
);
```

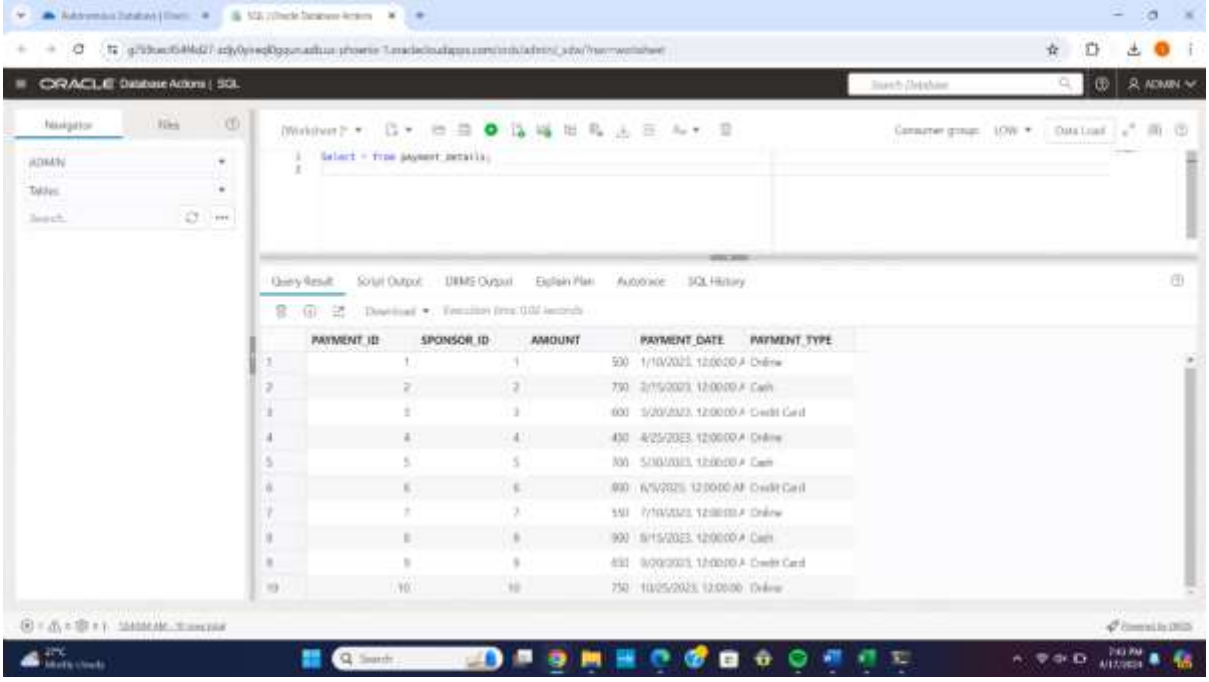
```
Table PAYMENT_DETAILS created.

Elapsed: 00:00:00.034
```

Inserting values to payment_details



```
INSERT INTO payment_details VALUES (1, 1, 500.00, DATE '2023-01-10', 'Online');
INSERT INTO payment_details VALUES (2, 2, 750.00, DATE '2023-02-15', 'Cash');
INSERT INTO payment_details VALUES (3, 3, 600.00, DATE '2023-03-20', 'Credit Card');
INSERT INTO payment_details VALUES (4, 4, 450.00, DATE '2023-04-25', 'Online');
INSERT INTO payment_details VALUES (5, 5, 700.00, DATE '2023-05-30', 'Cash');
INSERT INTO payment_details VALUES (6, 6, 800.00, DATE '2023-06-05', 'Credit Card');
INSERT INTO payment_details VALUES (7, 7, 550.00, DATE '2023-07-10', 'Online');
INSERT INTO payment_details VALUES (8, 8, 900.00, DATE '2023-08-15', 'Cash');
INSERT INTO payment_details VALUES (9, 9, 650.00, DATE '2023-09-20', 'Credit Card');
INSERT INTO payment_details VALUES (10, 10, 750.00, DATE '2023-10-25', 'Online');
```

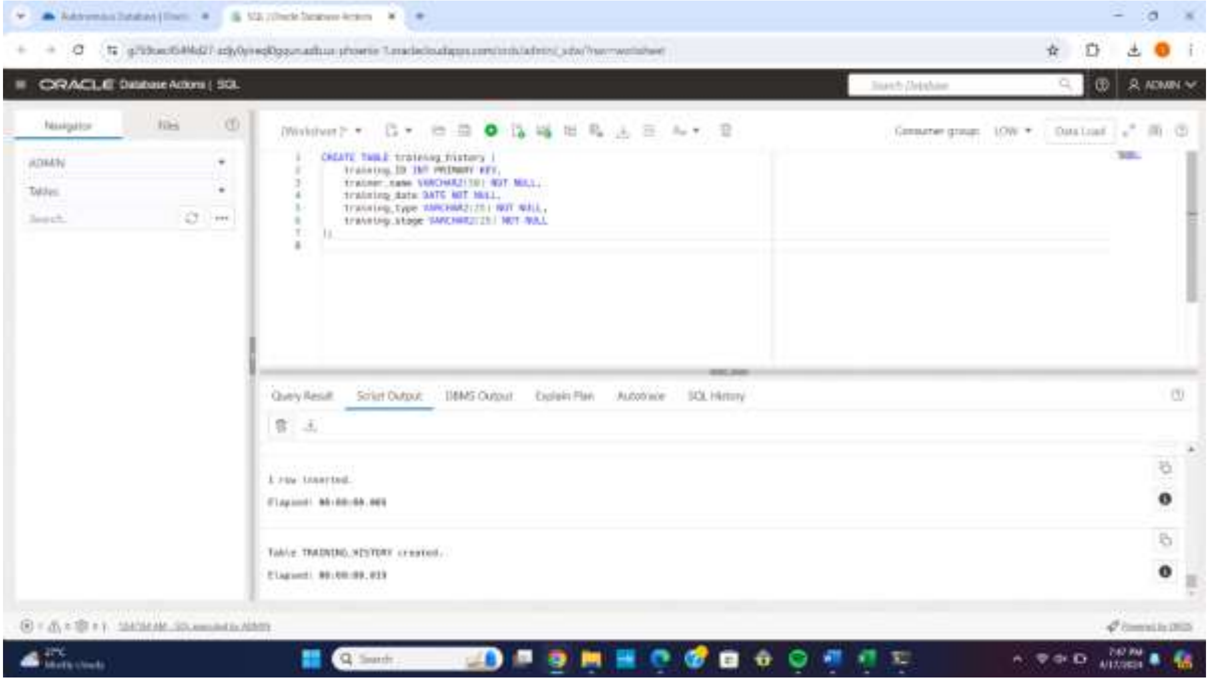
```
1 row inserted.

Elapsed: 00:00:00.003

1 row inserted.

Elapsed: 00:00:00.003
```



```
Select * from payment_details;
```

	PAYMENT_ID	SPONSOR_ID	AMOUNT	PAYMENT_DATE	PAYMENT_TYPE
1	1	1	500	1/10/2023, 12:00:00 A	Online
2	2	2	750	2/15/2023, 12:00:00 A	Cash
3	3	3	600	3/20/2023, 12:00:00 A	Credit Card
4	4	4	450	4/25/2023, 12:00:00 A	Online
5	5	5	700	5/30/2023, 12:00:00 A	Cash
6	6	6	800	6/5/2023, 12:00:00 AM	Credit Card
7	7	7	550	7/10/2023, 12:00:00 A	Online
8	8	8	900	8/15/2023, 12:00:00 A	Cash
9	9	9	650	9/20/2023, 12:00:00 A	Credit Card
10	10	10	750	10/25/2023, 12:00:00	Online

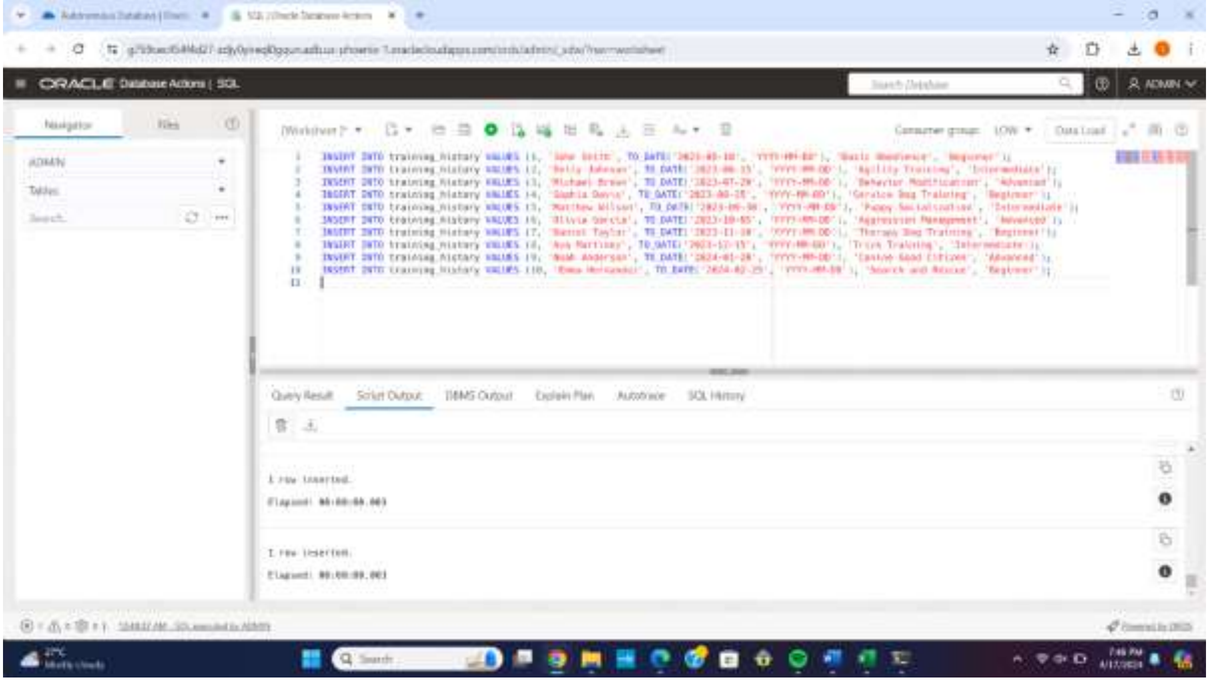
Creating a table named training_history



```
CREATE TABLE training_history (
    training_ID INT PRIMARY KEY,
    trainer_name VARCHAR2(30) NOT NULL,
    training_date DATE NOT NULL,
    training_type VARCHAR2(25) NOT NULL,
    training_stage VARCHAR2(25) NOT NULL
);
```

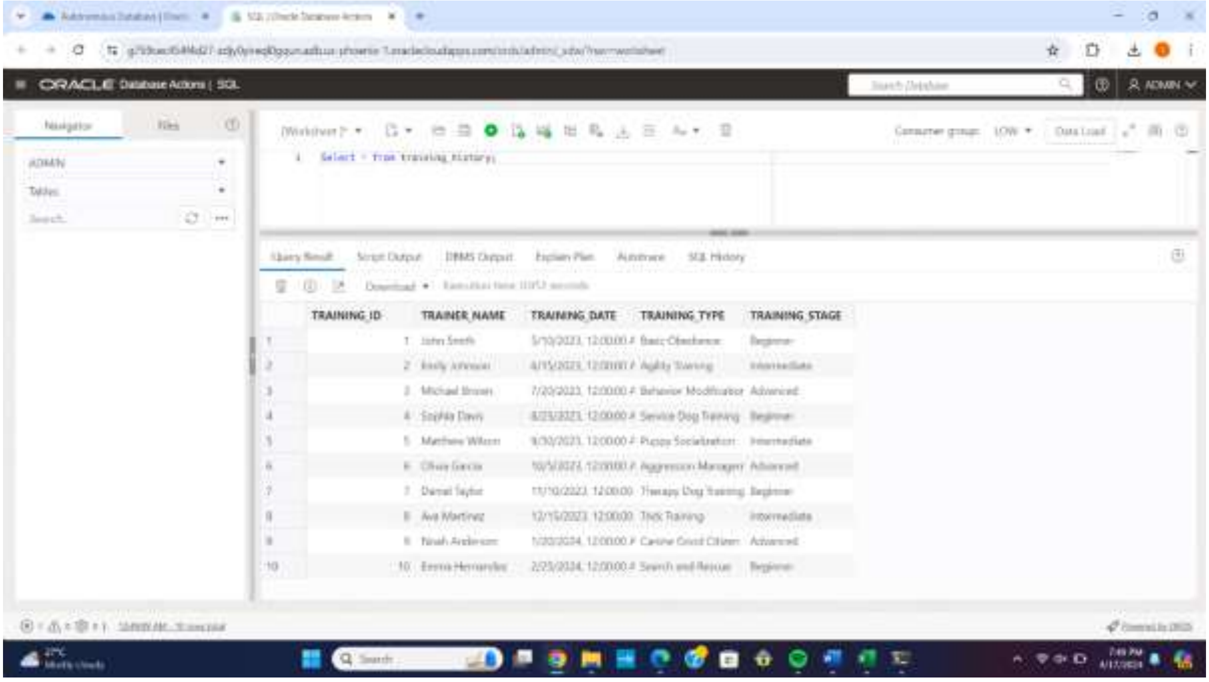
```
1 row inserted.

Elapsed: 00:00:00.003

Table TRAINING_HISTORY created.

Elapsed: 00:00:00.019
```

Inserting values to training_history



```
INSERT INTO training_history VALUES (1, 'John Smith', TO_DATE('2023-05-10', 'YYYY-MM-DD'), 'Basic Obedience', 'Beginner');
INSERT INTO training_history VALUES (2, 'Emily Johnson', TO_DATE('2023-06-15', 'YYYY-MM-DD'), 'Agility Training', 'Intermediate');
INSERT INTO training_history VALUES (3, 'Michael Brown', TO_DATE('2023-07-20', 'YYYY-MM-DD'), 'Behavior Modification', 'Advanced');
INSERT INTO training_history VALUES (4, 'Sophia Davis', TO_DATE('2023-08-25', 'YYYY-MM-DD'), 'Service Dog Training', 'Beginner');
INSERT INTO training_history VALUES (5, 'Matthew Wilson', TO_DATE('2023-09-30', 'YYYY-MM-DD'), 'Puppy Socialization', 'Intermediate');
INSERT INTO training_history VALUES (6, 'Olivia Garcia', TO_DATE('2023-10-05', 'YYYY-MM-DD'), 'Aggression Management', 'Advanced');
INSERT INTO training_history VALUES (7, 'Daniel Taylor', TO_DATE('2023-11-10', 'YYYY-MM-DD'), 'Therapy Dog Training', 'Beginner');
INSERT INTO training_history VALUES (8, 'Ava Martinez', TO_DATE('2023-12-15', 'YYYY-MM-DD'), 'Trick Training', 'Intermediate');
INSERT INTO training_history VALUES (9, 'Noah Anderson', TO_DATE('2024-01-20', 'YYYY-MM-DD'), 'Canine Good Citizen', 'Advanced');
INSERT INTO training_history VALUES (10, 'Emma Hernandez', TO_DATE('2024-02-25', 'YYYY-MM-DD'), 'Search and Rescue', 'Beginner');
```

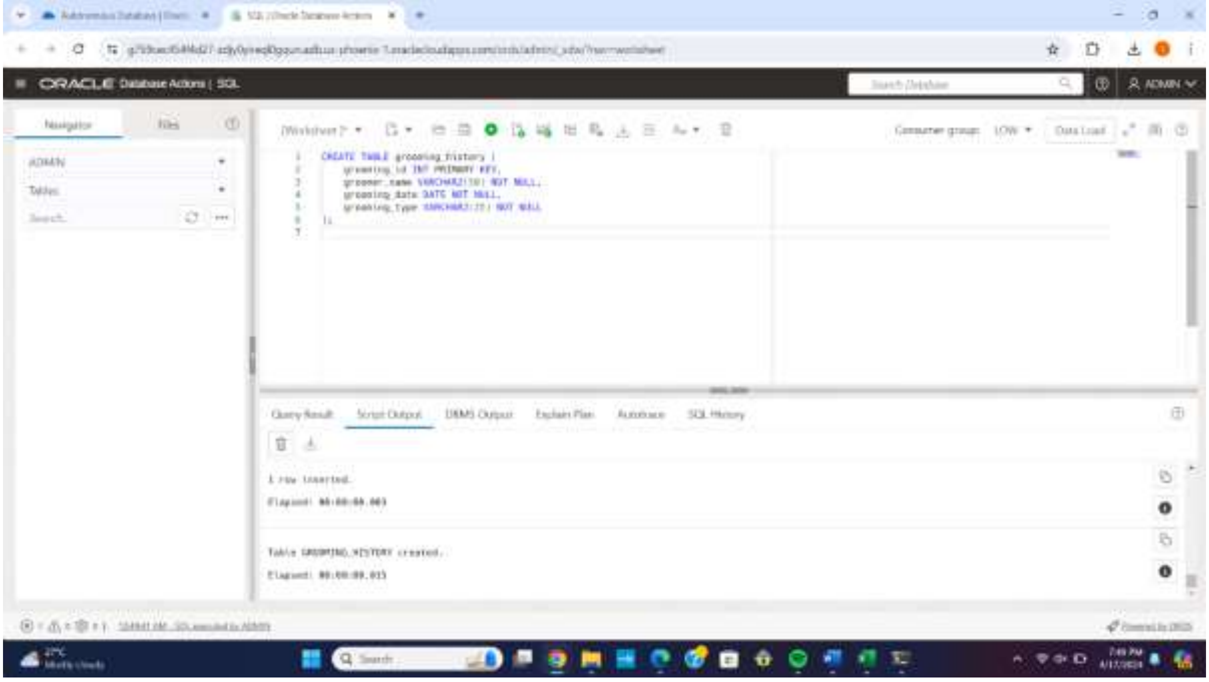
```
1 row inserted.

Elapsed: 00:00:00.003

1 row inserted.

Elapsed: 00:00:00.003
```



```sql
Select * from training_history;
```

	TRAINING_ID	TRAINER_NAME	TRAINING_DATE	TRAINING_TYPE	TRAINING_STAGE
1	1	John Smith	5/10/2023, 12:00:00 A	Basic Obedience	Beginner
2	2	Emily Johnson	6/15/2023, 12:00:00 A	Agility Training	Intermediate
3	3	Michael Brown	7/20/2023, 12:00:00 A	Behavior Modification	Advanced
4	4	Sophia Davis	8/25/2023, 12:00:00 A	Service Dog Training	Beginner
5	5	Matthew Wilson	9/30/2023, 12:00:00 A	Puppy Socialization	Intermediate
6	6	Olivia Garcia	10/5/2023, 12:00:00 A	Aggression Managem	Advanced
7	7	Daniel Taylor	11/10/2023, 12:00:00	Therapy Dog Training	Beginner
8	8	Ava Martinez	12/15/2023, 12:00:00	Trick Training	Intermediate
9	9	Noah Anderson	1/20/2024, 12:00:00 A	Canine Good Citizen	Advanced
10	10	Emma Hernandez	2/25/2024, 12:00:00 A	Search and Rescue	Beginner

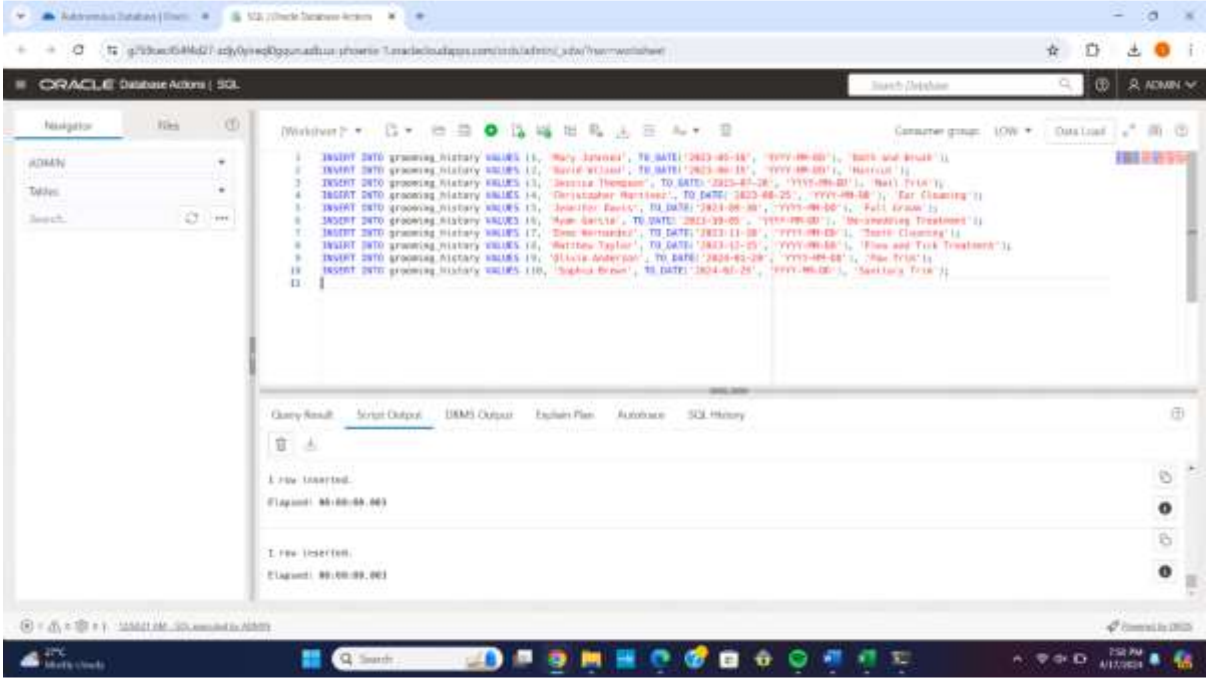
Creating a table named grooming_history



```sql
CREATE TABLE grooming_history (
    grooming_id INT PRIMARY KEY,
    groomer_name VARCHAR2(50) NOT NULL,
    grooming_date DATE NOT NULL,
    grooming_type VARCHAR2(25) NOT NULL
);
```

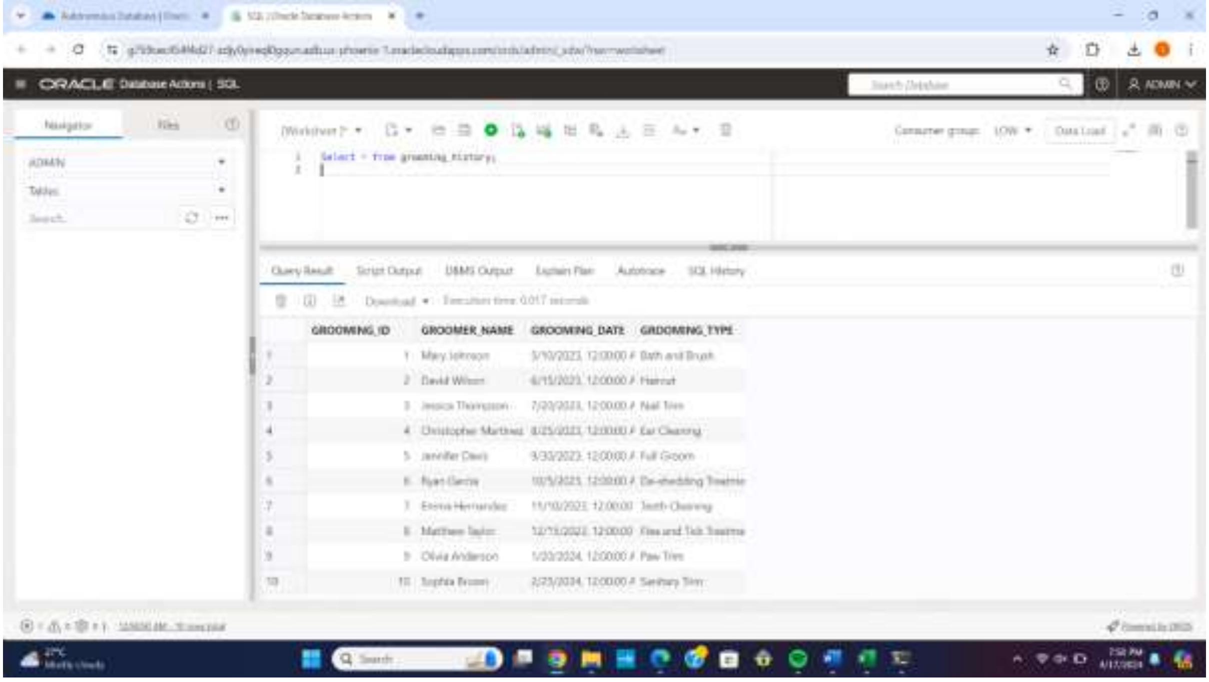
```
1 row inserted.

Elapsed: 00:00:00.003

Table GROOMING_HISTORY created.

Elapsed: 00:00:00.015
```

Inserting values to grooming_history



```
INSERT INTO grooming_history VALUES (1, 'Mary Johnson', TO_DATE('2023-05-10', 'YYYY-MM-DD'), 'Bath and Brush');
INSERT INTO grooming_history VALUES (2, 'David Wilson', TO_DATE('2023-06-15', 'YYYY-MM-DD'), 'Haircut');
INSERT INTO grooming_history VALUES (3, 'Jessica Thompson', TO_DATE('2023-07-20', 'YYYY-MM-DD'), 'Nail Trim');
INSERT INTO grooming_history VALUES (4, 'Christopher Martinez', TO_DATE('2023-08-25', 'YYYY-MM-DD'), 'Ear Cleaning');
INSERT INTO grooming_history VALUES (5, 'Jennifer Davis', TO_DATE('2023-09-30', 'YYYY-MM-DD'), 'Full Groom');
INSERT INTO grooming_history VALUES (6, 'Ryan Garcia', TO_DATE('2023-10-05', 'YYYY-MM-DD'), 'De-shedding Treatment');
INSERT INTO grooming_history VALUES (7, 'Emma Hernandez', TO_DATE('2023-11-10', 'YYYY-MM-DD'), 'Teeth Cleaning');
INSERT INTO grooming_history VALUES (8, 'Matthew Taylor', TO_DATE('2023-12-15', 'YYYY-MM-DD'), 'Flea and Tick Treatment');
INSERT INTO grooming_history VALUES (9, 'Olivia Anderson', TO_DATE('2024-01-20', 'YYYY-MM-DD'), 'Paw Trim');
INSERT INTO grooming_history VALUES (10, 'Sophia Brown', TO_DATE('2024-02-25', 'YYYY-MM-DD'), 'Sanitary Trim');
```

```
1 row inserted.

Elapsed: 00:00:00.003


1 row inserted.

Elapsed: 00:00:00.003
```

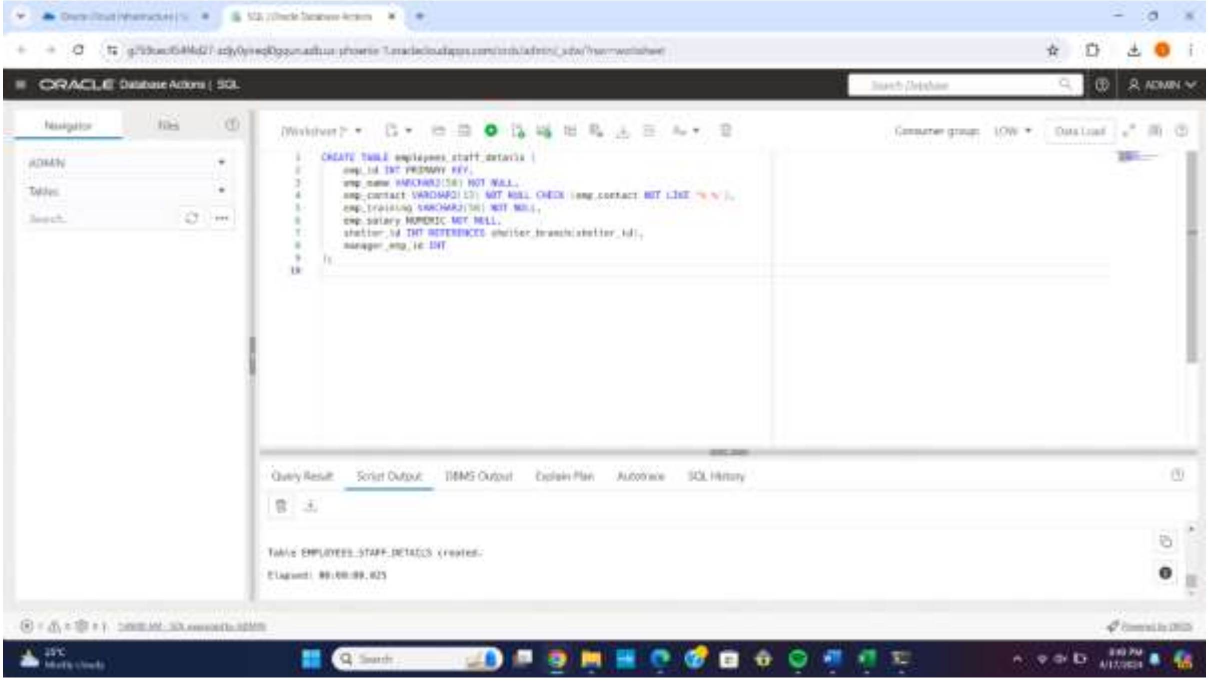


```
Select * from grooming_history;
```

Execution time: 0.017 seconds

	GROOMING_ID	GROOMER_NAME	GROOMING_DATE	GROOMING_TYPE
1	1	Mary Johnson	5/10/2023, 12:00:00 A	Bath and Brush
2	2	David Wilson	6/15/2023, 12:00:00 A	Haircut
3	3	Jessica Thompson	7/20/2023, 12:00:00 A	Nail Trim
4	4	Christopher Martinez	8/25/2023, 12:00:00 A	Ear Cleaning
5	5	Jennifer Davis	9/30/2023, 12:00:00 A	Full Groom
6	6	Ryan Garcia	10/5/2023, 12:00:00 A	De-shedding Treatme
7	7	Emma Hernandez	11/10/2023, 12:00:00	Teeth Cleaning
8	8	Matthew Taylor	12/15/2023, 12:00:00	Flea and Tick Treatme
9	9	Olivia Anderson	1/20/2024, 12:00:00 A	Paw Trim
10	10	Sophia Brown	2/25/2024, 12:00:00 A	Sanitary Trim

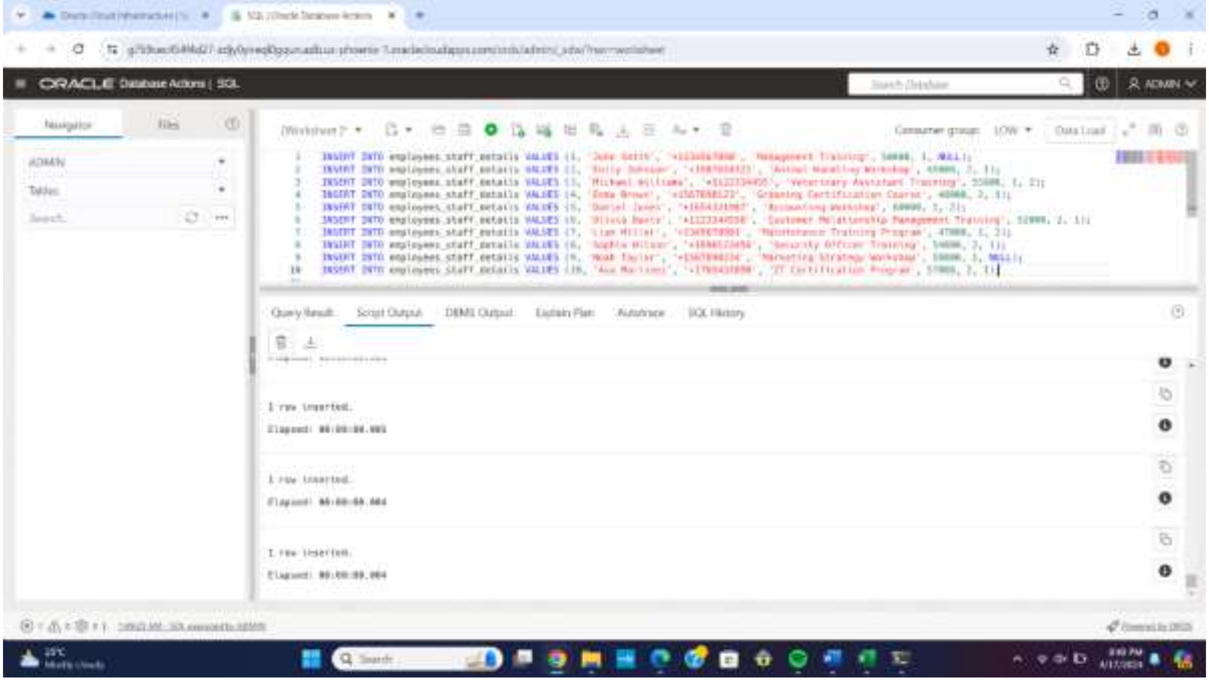
Creating a table named employees_staff_details



```
CREATE TABLE employees_staff_details (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR2(50) NOT NULL,
    emp_contact VARCHAR2(17) NOT NULL CHECK (emp_contact NOT LIKE '% %'),
    emp_training VARCHAR2(50) NOT NULL,
    emp_salary NUMERIC NOT NULL,
    shelter_id INT REFERENCES shelter_branch(shelter_id),
    manager_emp_id INT
);
```

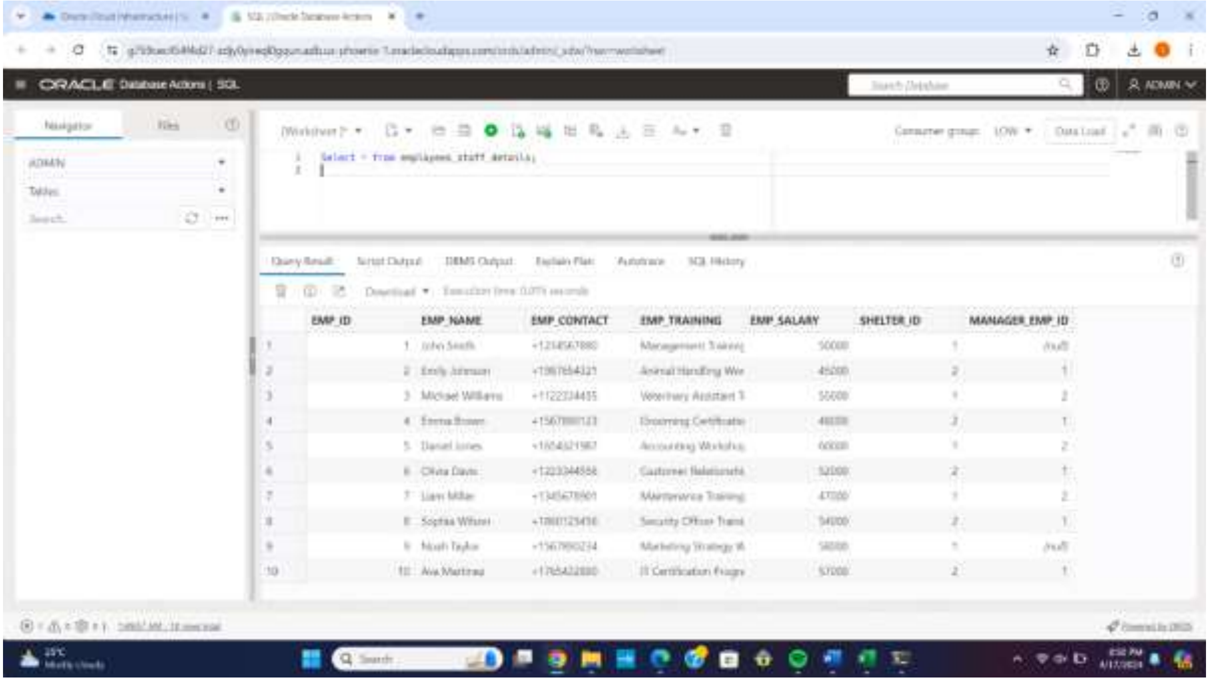
```
Table EMPLOYEES_STAFF_DETAILS created.

Elapsed: 00:00:00.025
```

Inserting values to employees_staff_details



```
INSERT INTO employees_staff_details VALUES (1, 'John Smith', '+1234567890', 'Management Training', 50000, 1, NULL);
INSERT INTO employees_staff_details VALUES (2, 'Emily Johnson', '+1987654321', 'Animal Handling Workshop', 45000, 2, 1);
INSERT INTO employees_staff_details VALUES (3, 'Michael Williams', '+1122334455', 'Veterinary Assistant Training', 55000, 1, 1);
INSERT INTO employees_staff_details VALUES (4, 'Emma Brown', '+1567890123', 'Grooming Certification Course', 40000, 2, 1);
INSERT INTO employees_staff_details VALUES (5, 'Daniel Jones', '+1654321987', 'Accounting Workshop', 60000, 1, 2);
INSERT INTO employees_staff_details VALUES (6, 'Olivia Davis', '+1122344556', 'Customer Relationship Management Training', 52000, 2, 1);
INSERT INTO employees_staff_details VALUES (7, 'Liam Miller', '+1345678901', 'Maintenance Training Program', 47000, 1, 2);
INSERT INTO employees_staff_details VALUES (8, 'Sophia Wilson', '+1890123456', 'Security Officer Training', 54000, 2, 1);
INSERT INTO employees_staff_details VALUES (9, 'Noah Taylor', '+1567890234', 'Marketing Strategy Workshop', 58000, 1, NULL);
INSERT INTO employees_staff_details VALUES (10, 'Ava Martinez', '+1765432890', 'IT Certification Program', 57000, 2, 1);
```

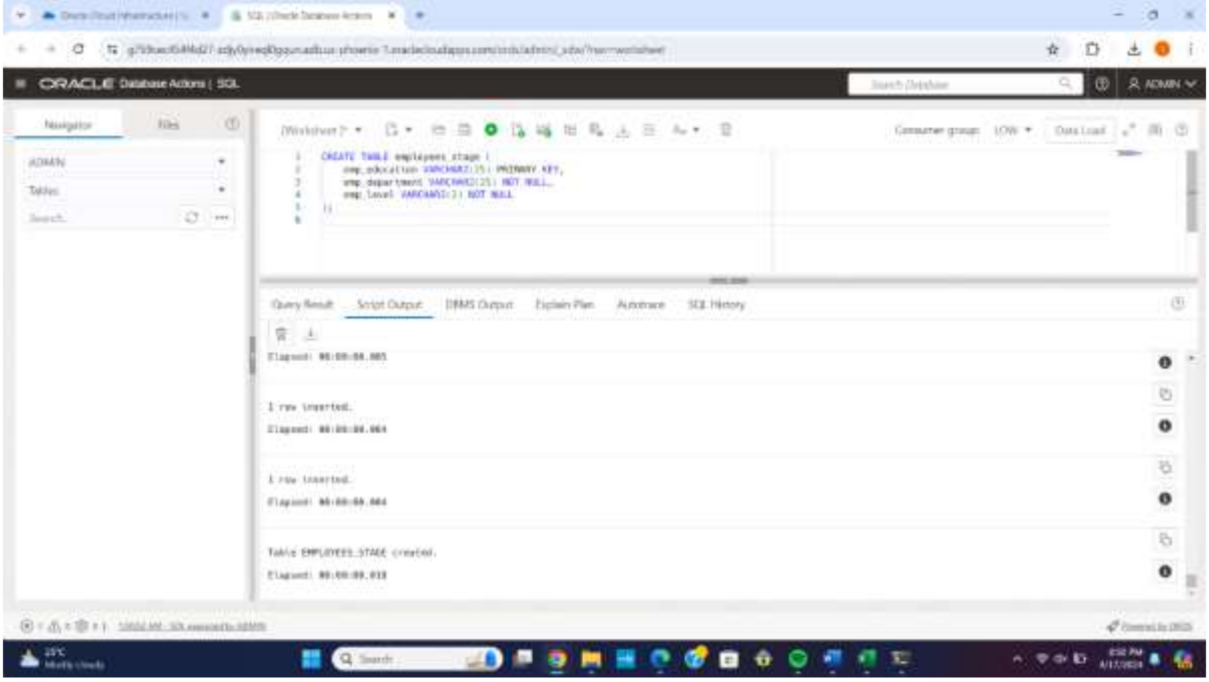
```
1 row inserted.

Elapsed: 00:00:00.005

1 row inserted.

Elapsed: 00:00:00.004

1 row inserted.

Elapsed: 00:00:00.004
```



```
Select * from employees_staff_details;
```

	EMP_ID	EMP_NAME	EMP_CONTACT	EMP_TRAINING	EMP_SALARY	SHELTER_ID	MANAGER_EMP_ID
1	1	John Smith	+1234567890	Management Training	50000	1	(null)
2	2	Emily Johnson	+1987654321	Animal Handling Wor	45000	2	1
3	3	Michael Williams	+1122334455	Veterinary Assistant T	55000	1	2
4	4	Emma Brown	+1567890123	Grooming Certificatio	48000	2	1
5	5	Daniel Jones	+1654321987	Accounting Worksho	60000	1	2
6	6	Olivia Davis	+1223344556	Customer Relationshi	52000	2	1
7	7	Liam Miller	+1345678901	Maintenance Training	47000	1	2
8	8	Sophia Wilson	+1890123456	Security Officer Traini	54000	2	1
9	9	Noah Taylor	+1567890234	Marketing Strategy W	58000	1	(null)
10	10	Ava Martinez	+1765432890	IT Certification Progr	57000	2	1

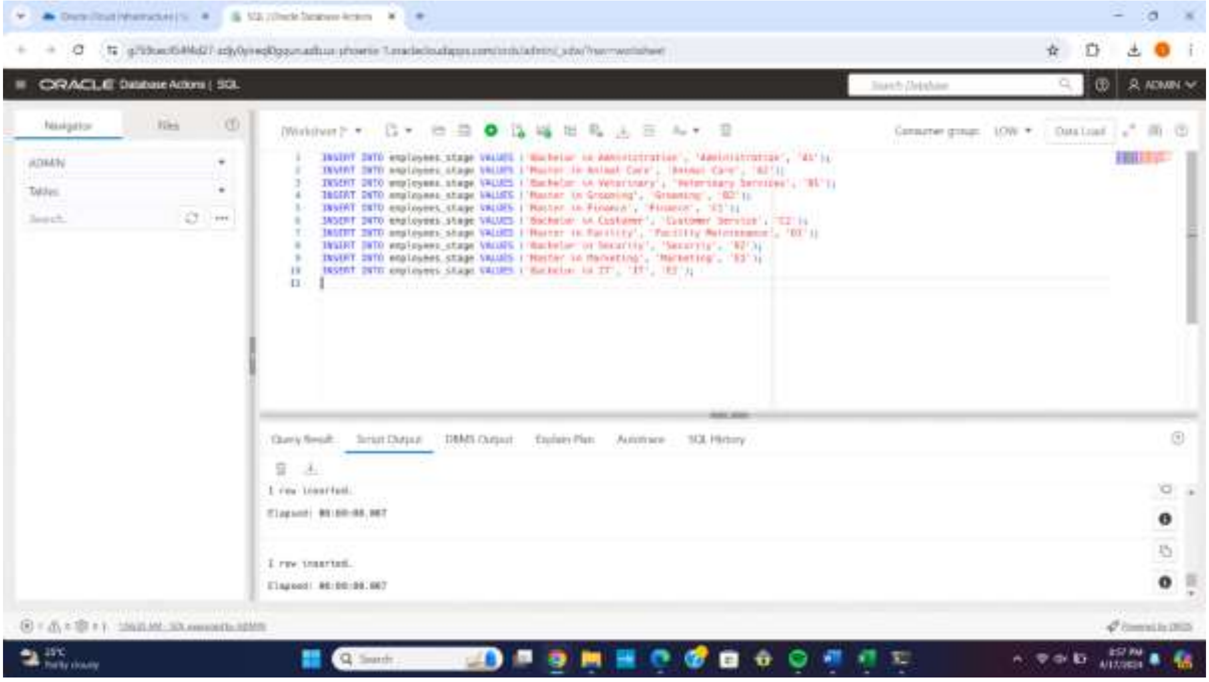
Creating a table named employees_stage(New)



```
CREATE TABLE employees_stage (
    emp_education VARCHAR2(15) PRIMARY KEY,
    emp_department VARCHAR2(25) NOT NULL,
    emp_level VARCHAR2(2) NOT NULL
);
```

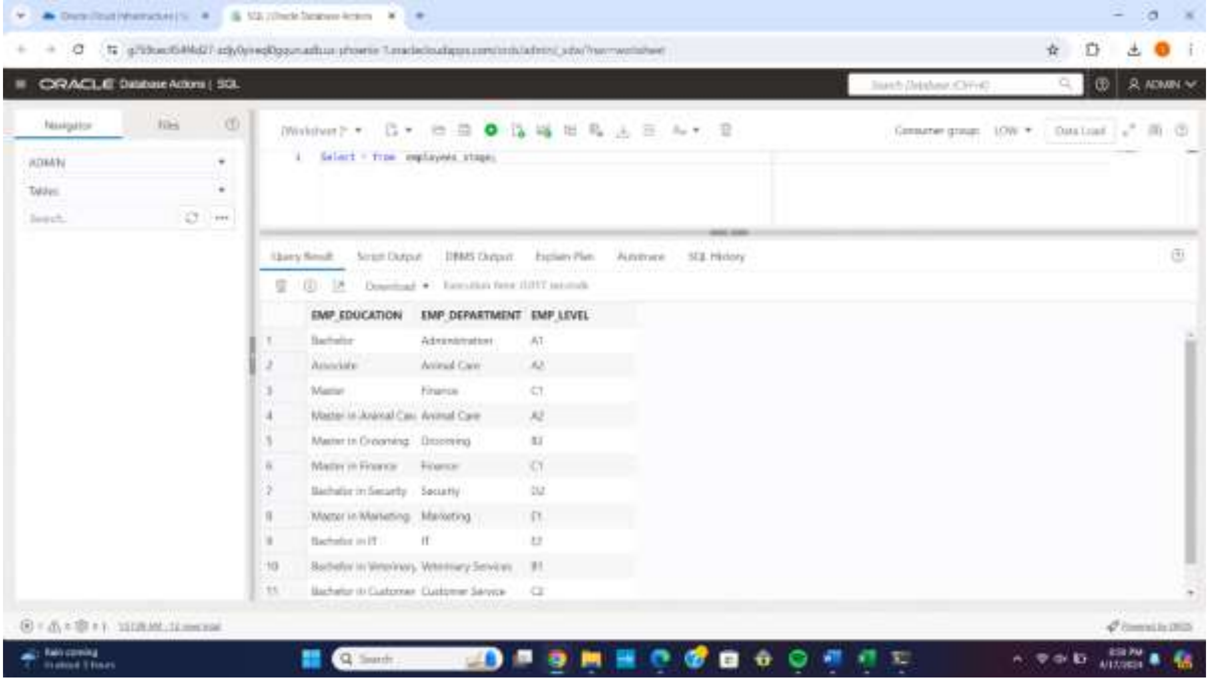
```
Elapsed: 00:00:00.005

1 row inserted.

Elapsed: 00:00:00.004

1 row inserted.

Elapsed: 00:00:00.004

Table EMPLOYEES_STAGE created.

Elapsed: 00:00:00.018
```

Inserting values to employees_stage(New)



```
INSERT INTO employees_stage VALUES ('Bachelor in Administration', 'Administration', 'A1');
INSERT INTO employees_stage VALUES ('Master in Animal Care', 'Animal Care', 'A2');
INSERT INTO employees_stage VALUES ('Bachelor in Veterinary', 'Veterinary Services', 'B1');
INSERT INTO employees_stage VALUES ('Master in Grooming', 'Grooming', 'B2');
INSERT INTO employees_stage VALUES ('Master in Finance', 'Finance', 'C1');
INSERT INTO employees_stage VALUES ('Bachelor in Customer', 'Customer Service', 'C2');
INSERT INTO employees_stage VALUES ('Master in Facility', 'Facility Maintenance', 'D1');
INSERT INTO employees_stage VALUES ('Bachelor in Security', 'Security', 'D2');
INSERT INTO employees_stage VALUES ('Master in Marketing', 'Marketing', 'E1');
INSERT INTO employees_stage VALUES ('Bachelor in IT', 'IT', 'E2');
```

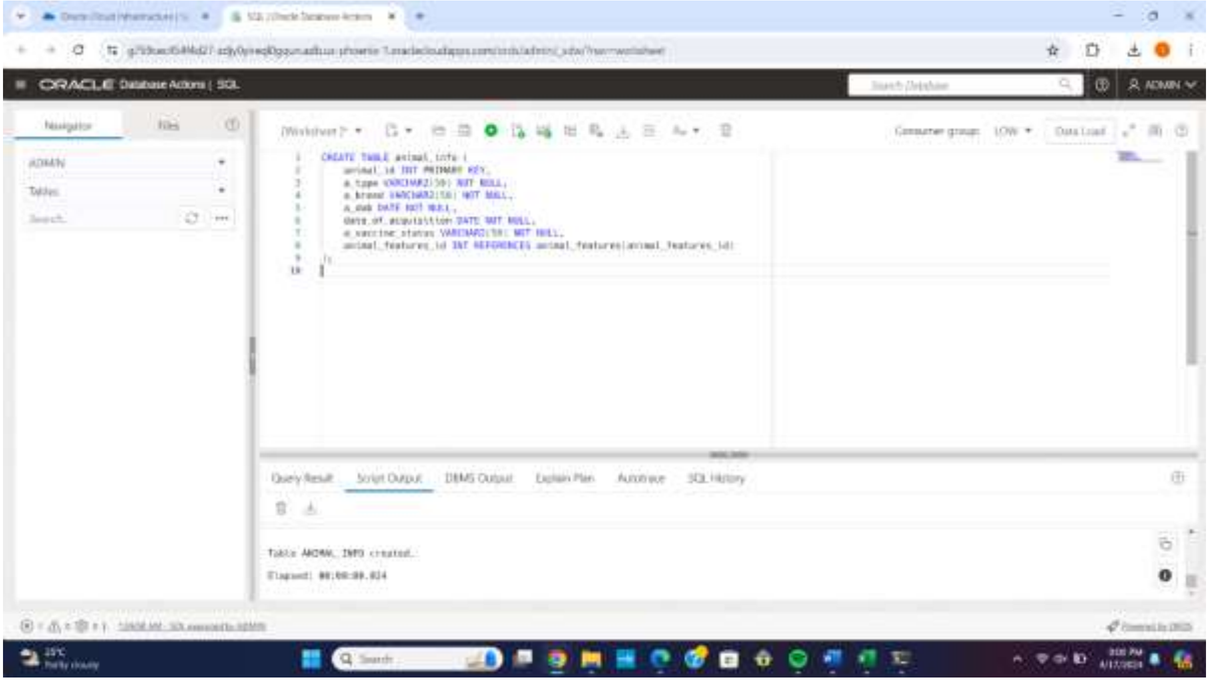
```
1 row inserted.

Elapsed: 00:00:00.007

1 row inserted.

Elapsed: 00:00:00.007
```



```
Select * from employees_stage;
```

	EMP_EDUCATION	EMP_DEPARTMENT	EMP_LEVEL
1	Bachelor	Administration	A1
2	Associate	Animal Care	A2
3	Master	Finance	C1
4	Master in Animal Car	Animal Care	A2
5	Master in Grooming	Grooming	B2
6	Master in Finance	Finance	C1
7	Bachelor in Security	Security	D2
8	Master in Marketing	Marketing	E1
9	Bachelor in IT	IT	E2
10	Bachelor in Veterinary	Veterinary Services	B1
11	Bachelor in Customer	Customer Service	C2

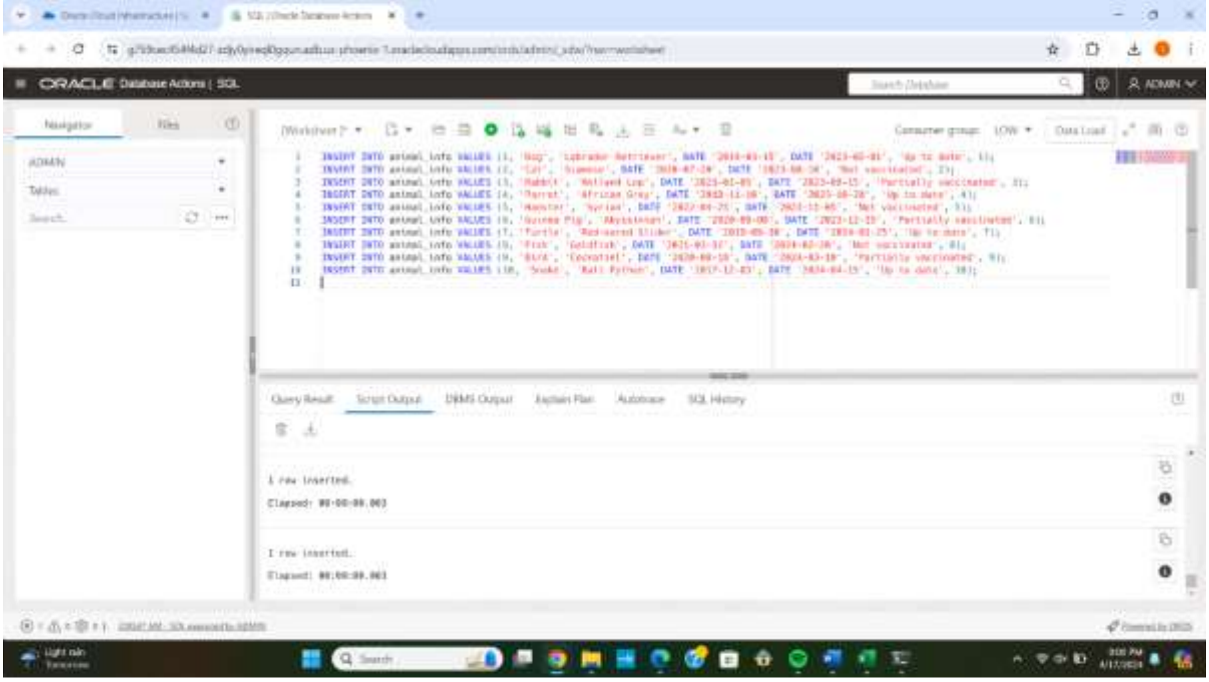
Creating a table named animal_info



```sql
CREATE TABLE animal_info (
    animal_id INT PRIMARY KEY,
    a_type VARCHAR2(50) NOT NULL,
    a_breed VARCHAR2(50) NOT NULL,
    a_dob DATE NOT NULL,
    date_of_acquisition DATE NOT NULL,
    a_vaccine_status VARCHAR2(50) NOT NULL,
    animal_features_id INT REFERENCES animal_features(animal_features_id)
);
```

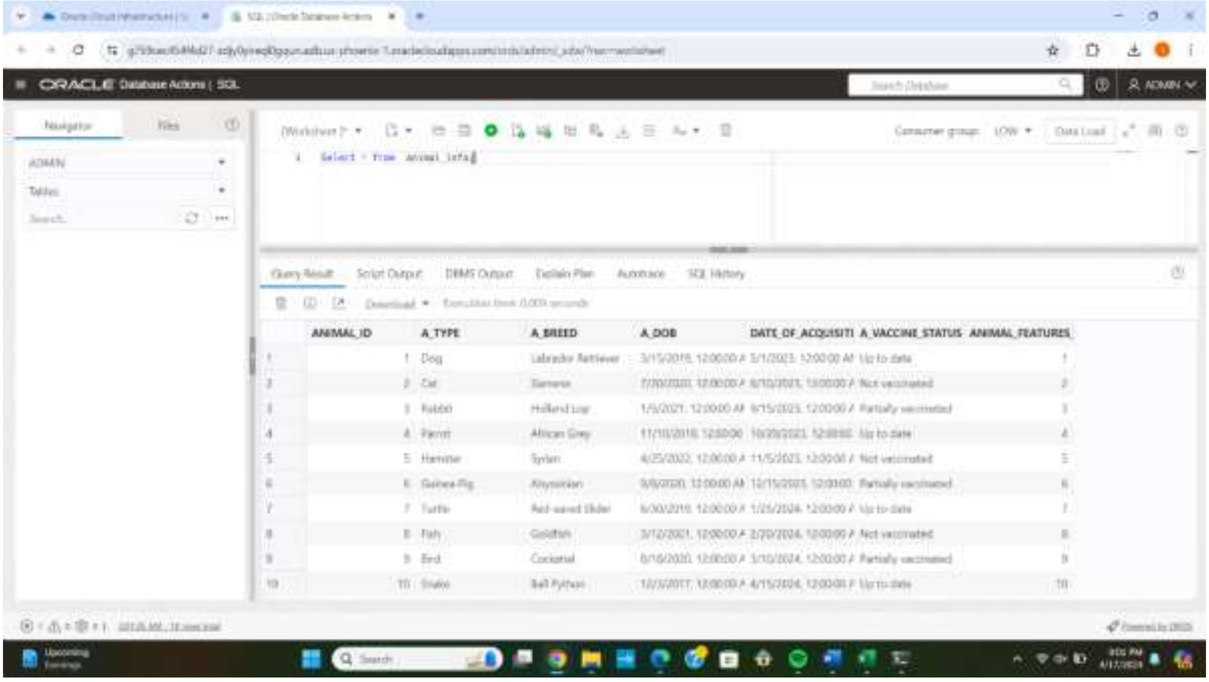
```
Table ANIMAL_INFO created.

Elapsed: 00:00:00.024
```

Inserting values to animal_info



```sql
INSERT INTO animal_info VALUES (1, 'Dog', 'Labrador Retriever', DATE '2018-03-15', DATE '2023-02-01', 'Up to date', 1);
INSERT INTO animal_info VALUES (2, 'Cat', 'Siamese', DATE '2019-07-20', DATE '2023-08-10', 'Not vaccinated', 2);
INSERT INTO animal_info VALUES (3, 'Rabbit', 'Holland Lop', DATE '2023-01-05', DATE '2023-09-15', 'Partially vaccinated', 3);
INSERT INTO animal_info VALUES (4, 'Parrot', 'African Grey', DATE '2022-11-10', DATE '2023-10-20', 'Up to date', 4);
INSERT INTO animal_info VALUES (5, 'Hamster', 'Syrian', DATE '2022-04-25', DATE '2023-11-05', 'Not vaccinated', 5);
INSERT INTO animal_info VALUES (6, 'Guinea Pig', 'Abyssinian', DATE '2020-09-08', DATE '2023-12-15', 'Partially vaccinated', 6);
INSERT INTO animal_info VALUES (7, 'Turtle', 'Red-eared Slider', DATE '2015-05-30', DATE '2024-01-25', 'Up to date', 7);
INSERT INTO animal_info VALUES (8, 'Fish', 'Goldfish', DATE '2021-03-12', DATE '2024-02-20', 'Not vaccinated', 8);
INSERT INTO animal_info VALUES (9, 'Bird', 'Cockatiel', DATE '2020-08-10', DATE '2024-03-10', 'Partially vaccinated', 9);
INSERT INTO animal_info VALUES (10, 'Snake', 'Ball Python', DATE '2017-12-03', DATE '2024-04-15', 'Up to date', 10);
```

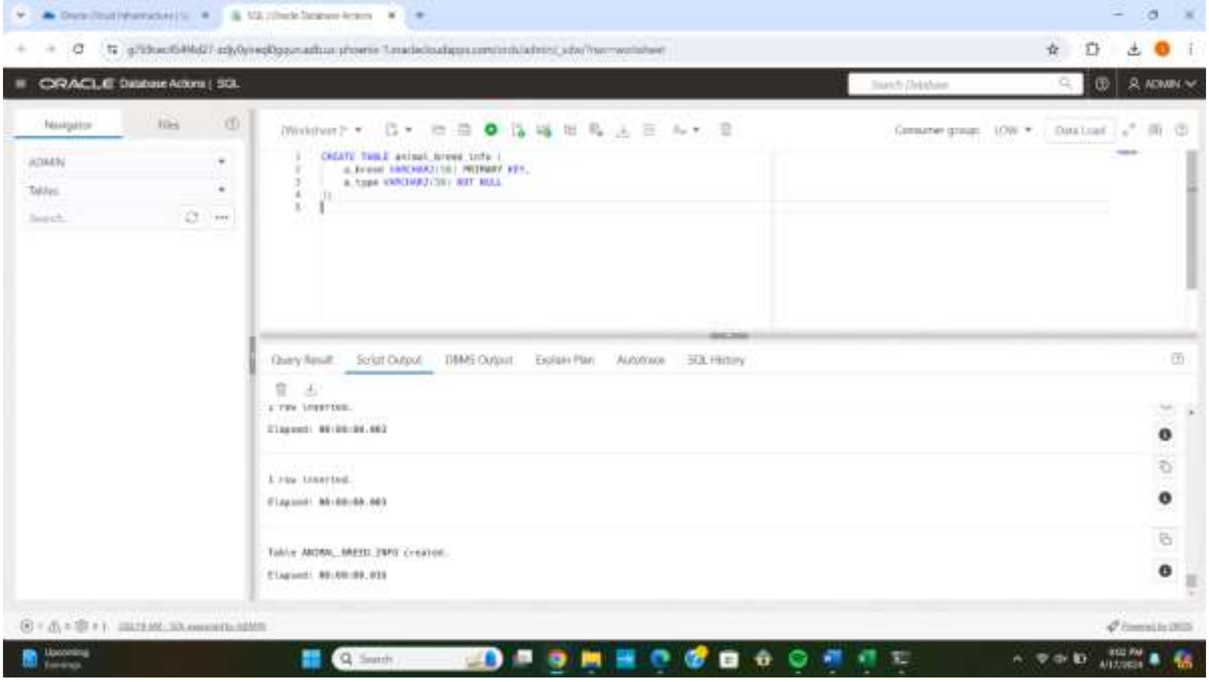
```
1 row inserted.

Elapsed: 00:00:00.003

1 row inserted.

Elapsed: 00:00:00.003
```



```
Select * from animal_info;
```

	ANIMAL_ID	A_TYPE	A_BREED	A_DOB	DATE_OF_ACQUISITI	A_VACCINE_STATUS	ANIMAL_FEATURES
1	1	Dog	Labrador Retriever	3/15/2019, 12:00:00 A	5/1/2023, 12:00:00 AI	Up to date	1
2	2	Cat	Siamese	7/30/2020, 12:00:00 A	8/10/2023, 12:00:00 A	Not vaccinated	2
3	3	Rabbit	Holland Lop	1/5/2021, 12:00:00 AI	9/15/2023, 12:00:00 A	Partially vaccinated	3
4	4	Parrot	African Grey	11/10/2018, 12:00:00	10/20/2023, 12:00:00	Up to date	4
5	5	Hamster	Syrian	4/25/2022, 12:00:00 A	11/5/2023, 12:00:00 A	Not vaccinated	5
6	6	Guinea Pig	Abyssinian	9/8/2020, 12:00:00 AI	12/15/2023, 12:00:00	Partially vaccinated	6
7	7	Turtle	Red-eared Slider	6/30/2019, 12:00:00 A	1/25/2024, 12:00:00 A	Up to date	7
8	8	Fish	Goldfish	3/12/2021, 12:00:00 A	2/20/2024, 12:00:00 A	Not vaccinated	8
9	9	Bird	Cockatiel	8/18/2020, 12:00:00 A	3/10/2024, 12:00:00 A	Partially vaccinated	9
10	10	Snake	Ball Python	12/3/2017, 12:00:00 A	4/15/2024, 12:00:00 F	Up to date	10

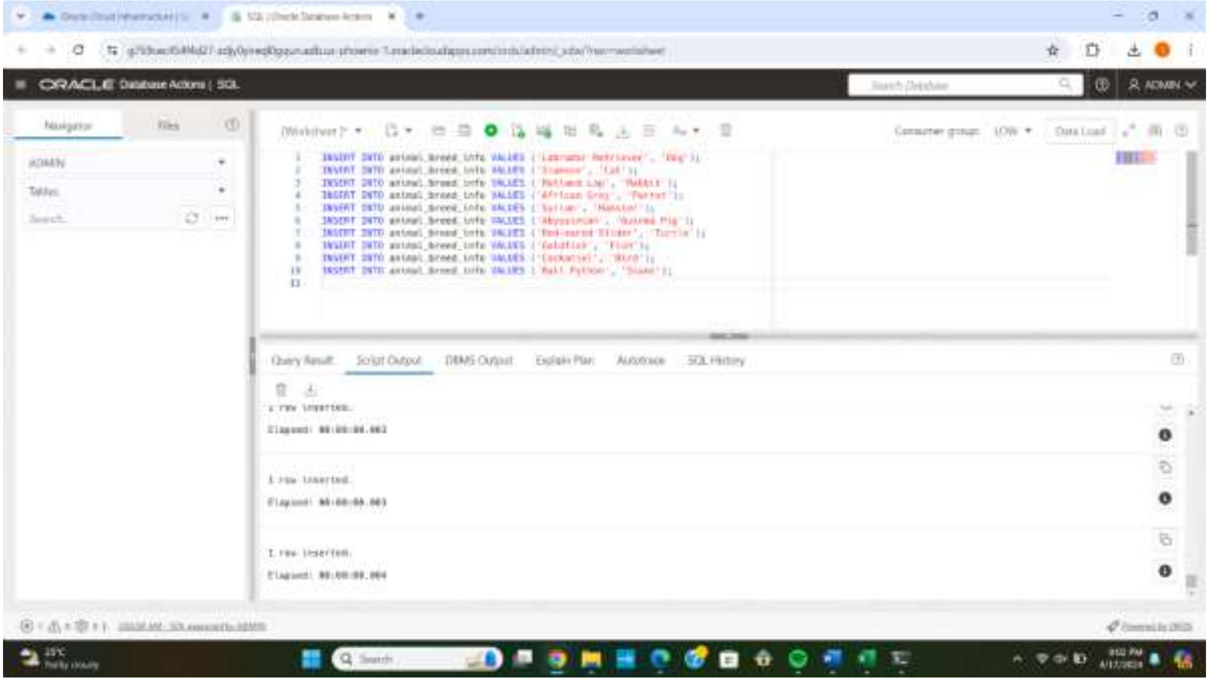
Creating a table named animal_breed_info(New)



```
CREATE TABLE animal_breed_info (
    a_breed VARCHAR2(50) PRIMARY KEY,
    a_type VARCHAR2(50) NOT NULL
);
```

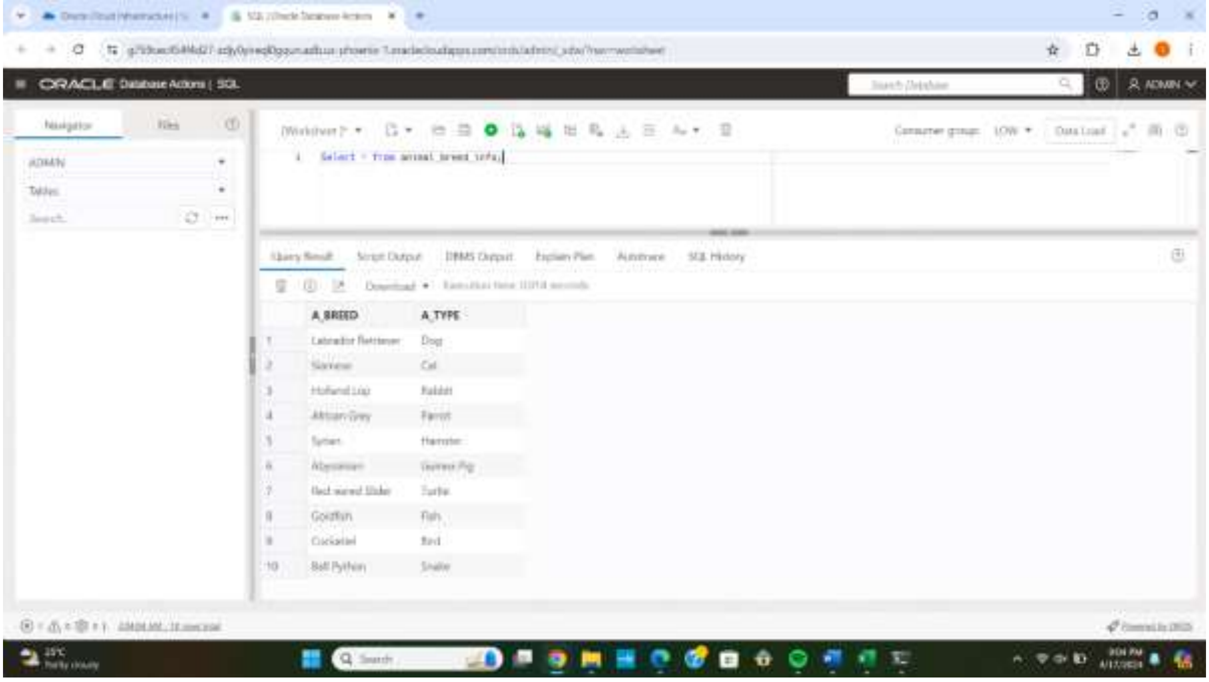
```
1 row inserted.

Elapsed: 00:00:00.002


1 row inserted.

Elapsed: 00:00:00.003


Table ANIMAL_BREED_INFO created.

Elapsed: 00:00:00.018
```

Inserting values to animal_breed_info(New)



```
INSERT INTO animal_breed_info VALUES ('Labrador Retriever', 'Dog');
INSERT INTO animal_breed_info VALUES ('Siamese', 'Cat');
INSERT INTO animal_breed_info VALUES ('Holland Lop', 'Rabbit');
INSERT INTO animal_breed_info VALUES ('African Grey', 'Parrot');
INSERT INTO animal_breed_info VALUES ('Syrian', 'Hamster');
INSERT INTO animal_breed_info VALUES ('Abyssinian', 'Guinea Pig');
INSERT INTO animal_breed_info VALUES ('Red-eared Slider', 'Turtle');
INSERT INTO animal_breed_info VALUES ('Goldfish', 'Fish');
INSERT INTO animal_breed_info VALUES ('Cockatiel', 'Bird');
INSERT INTO animal_breed_info VALUES ('Ball Python', 'Snake');
```

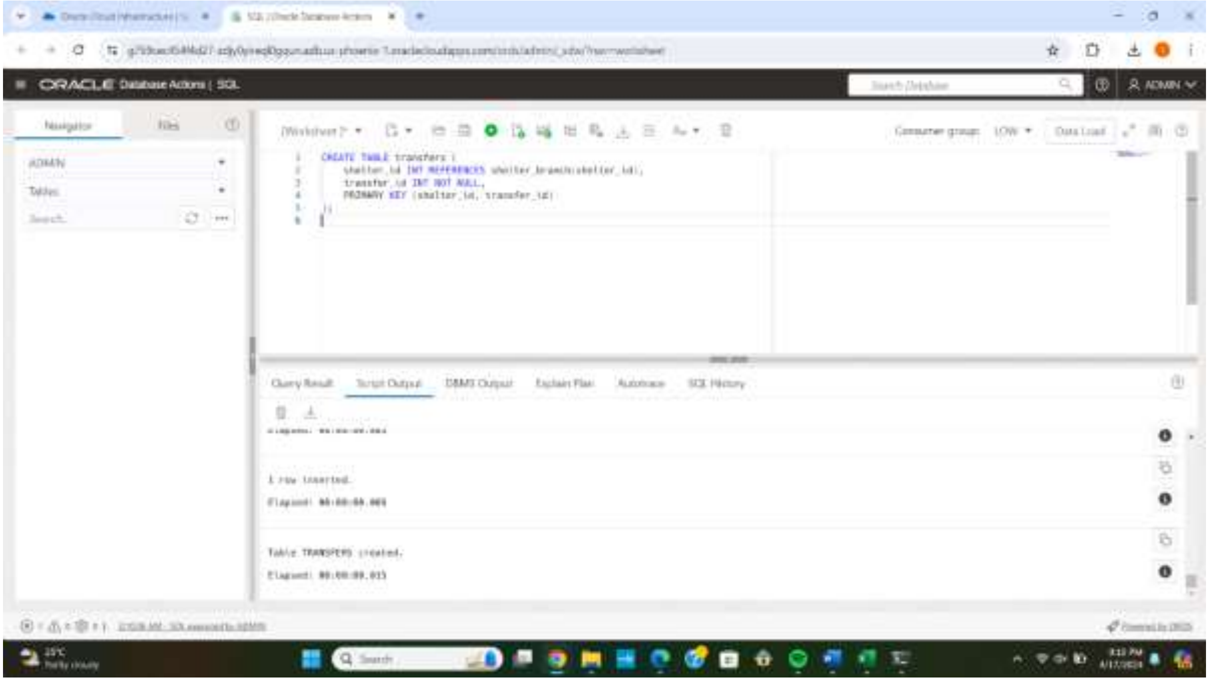
```
1 row inserted.

Elapsed: 00:00:00.003


1 row inserted.

Elapsed: 00:00:00.003


1 row inserted.

Elapsed: 00:00:00.004
```



```
Select * from animal_breed_info;
```

	A_BREED	A_TYPE
1	Labrador Retriever	Dog
2	Siamese	Cat
3	Holland Lop	Rabbit
4	African Grey	Parrot
5	Syrian	Hamster
6	Abyssinian	Guinea Pig
7	Red-eared Slider	Turtle
8	Goldfish	Fish
9	Cockatiel	Bird
10	Ball Python	Snake

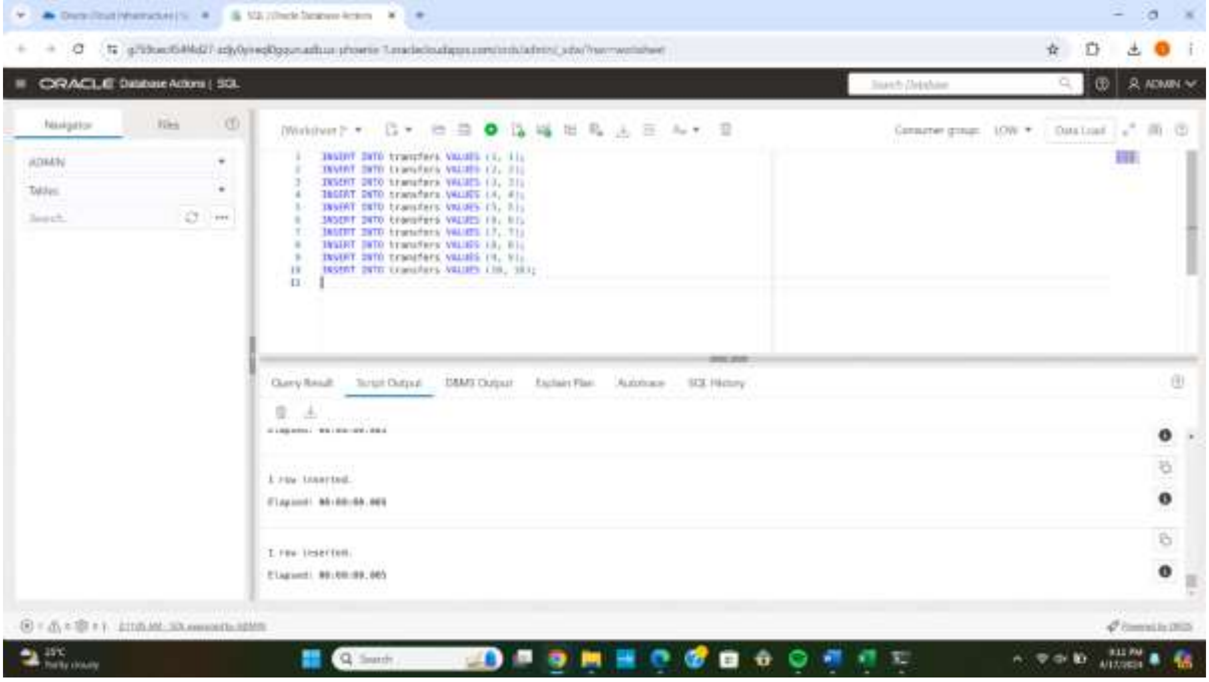
Creating a table named transfers



```
CREATE TABLE transfers (
    shelter_id INT REFERENCES shelter_branch(shelter_id),
    transfer_id INT NOT NULL,
    PRIMARY KEY (shelter_id, transfer_id)
);
```

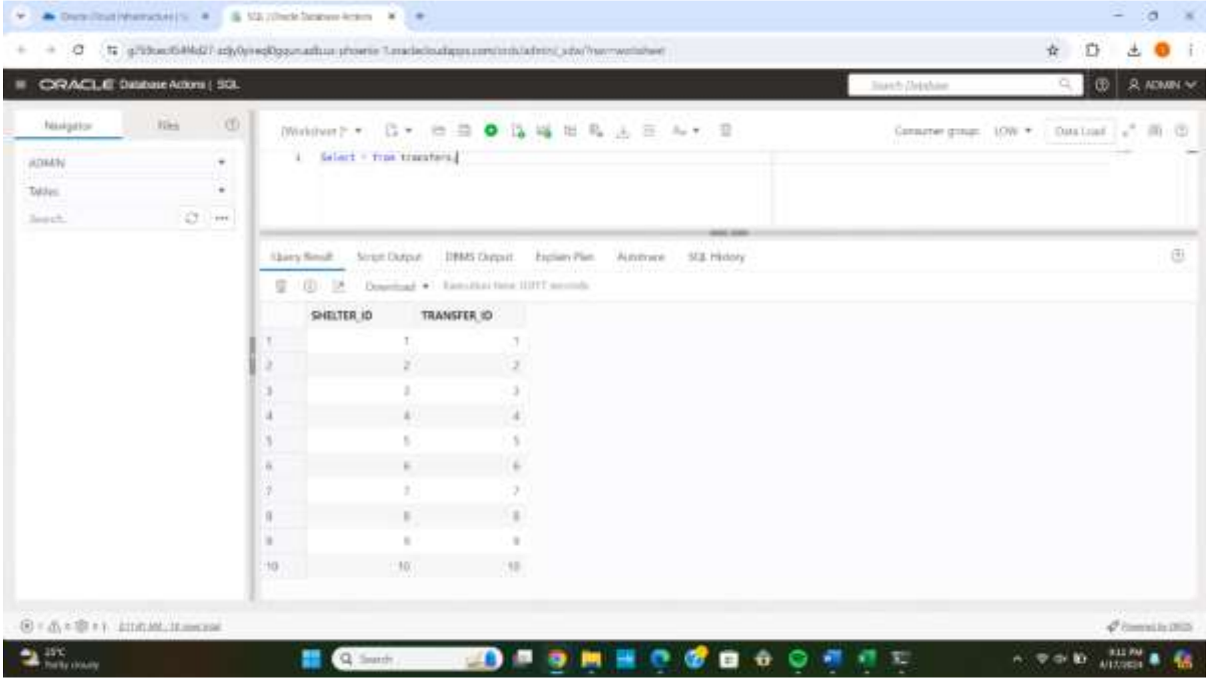
```
1 row inserted.

Elapsed: 00:00:00.003

Table TRANSFERS created.

Elapsed: 00:00:00.015
```

Inserting values to transfers



```
INSERT INTO transfers VALUES (1, 1);
INSERT INTO transfers VALUES (2, 2);
INSERT INTO transfers VALUES (3, 3);
INSERT INTO transfers VALUES (4, 4);
INSERT INTO transfers VALUES (5, 5);
INSERT INTO transfers VALUES (6, 6);
INSERT INTO transfers VALUES (7, 7);
INSERT INTO transfers VALUES (8, 8);
INSERT INTO transfers VALUES (9, 9);
INSERT INTO transfers VALUES (10, 10);
```

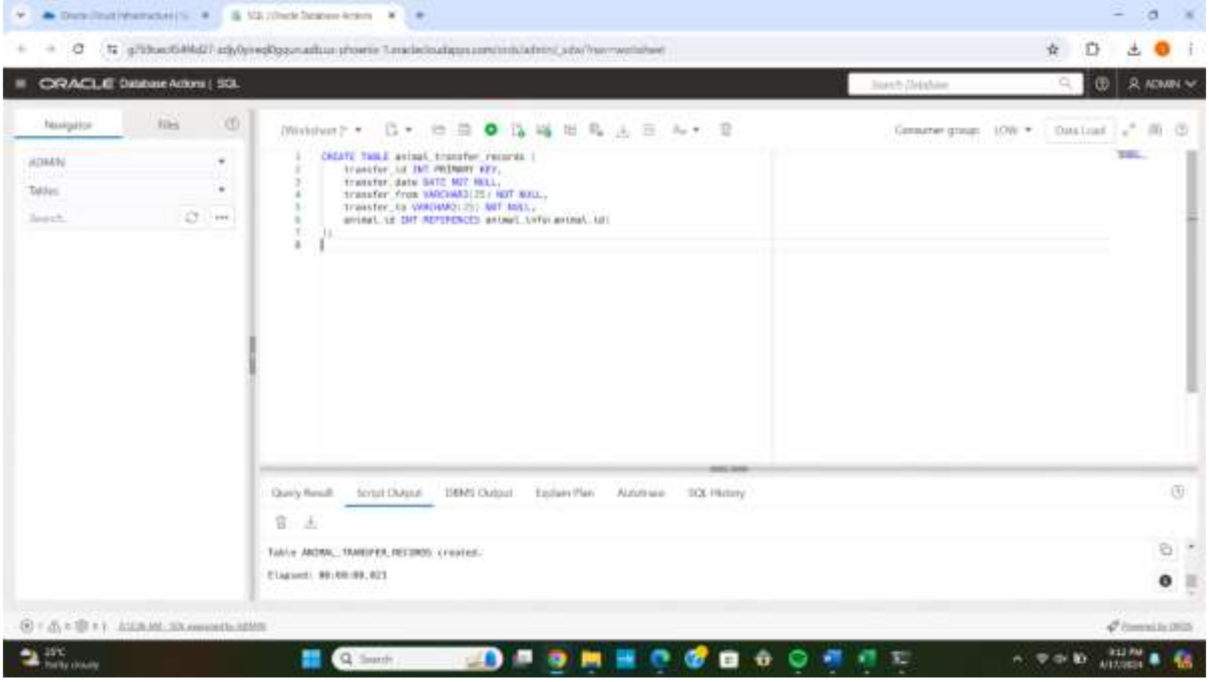
```
1 row inserted.

Elapsed: 00:00:00.003

1 row inserted.

Elapsed: 00:00:00.005
```



```
Select * from transfers;
```

	SHELTER_ID	TRANSFER_ID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10

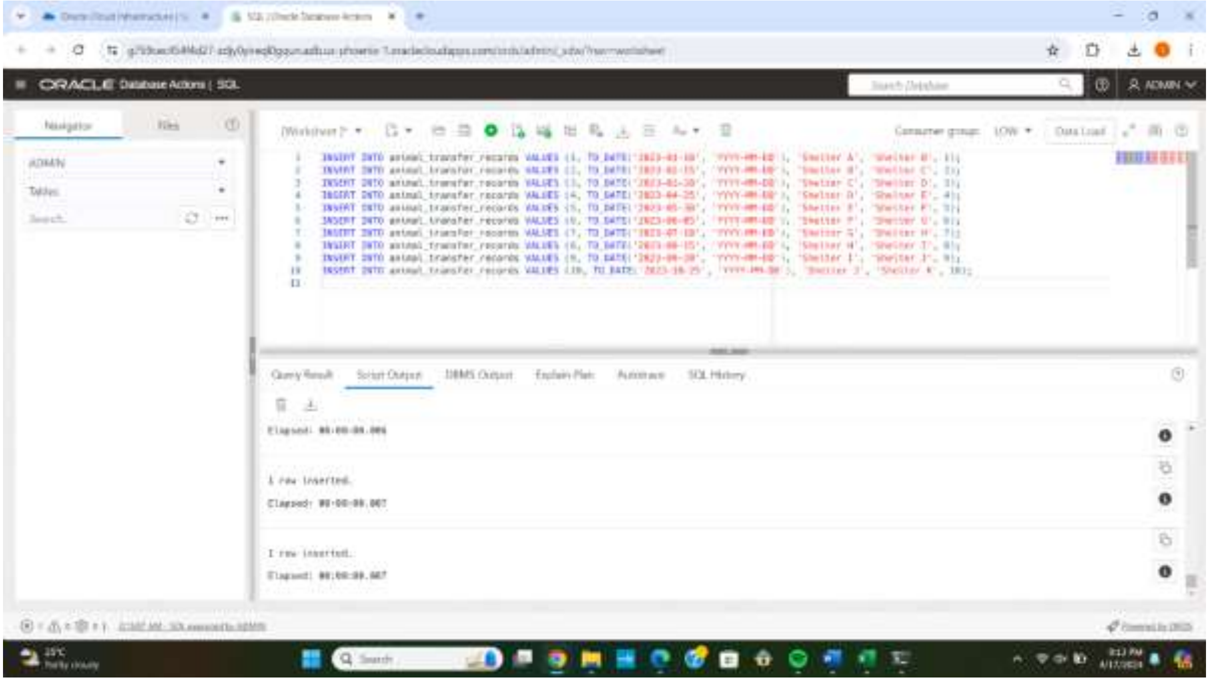
Creating a table named animal_transfer_records



```
CREATE TABLE animal_transfer_records (
    transfer_id INT PRIMARY KEY,
    transfer_date DATE NOT NULL,
    transfer_from VARCHAR2(25) NOT NULL,
    transfer_to VARCHAR2(25) NOT NULL,
    animal_id INT REFERENCES animal_info(animal_id)
);
```

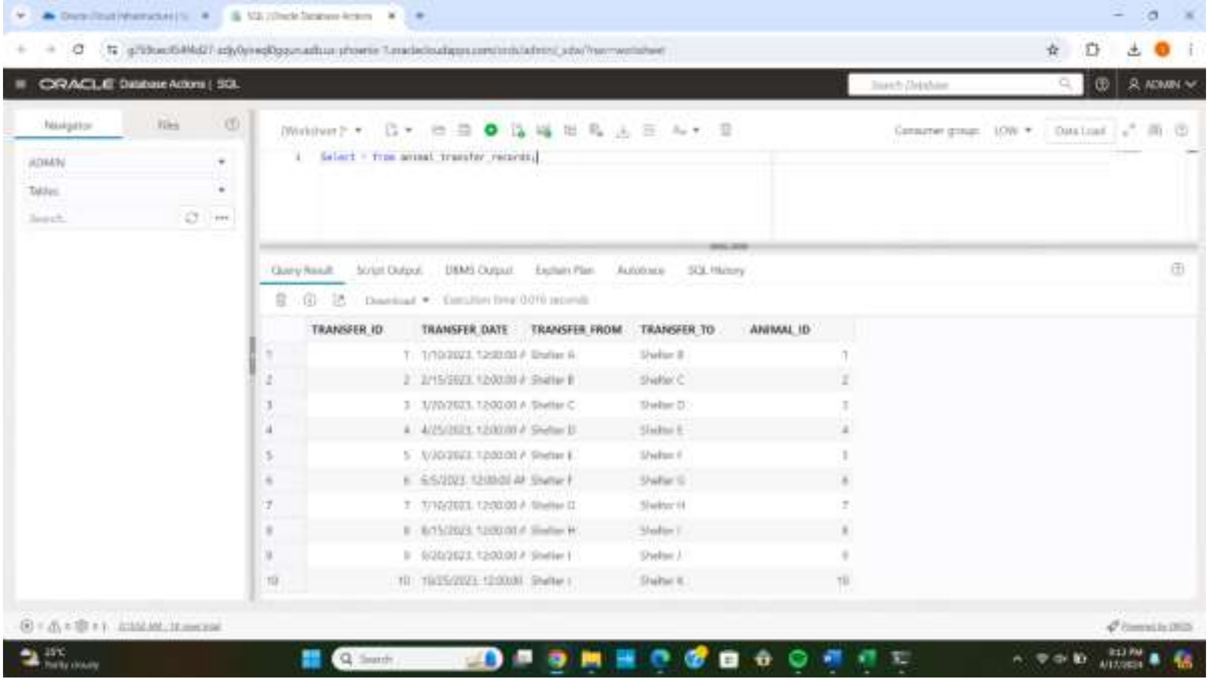
```
Table ANIMAL_TRANSFER_RECORDS created.

Elapsed: 00:00:00.023
```

Inserting values to animal_transfer_records



```
INSERT INTO animal_transfer_records VALUES (1, TO_DATE('2023-01-10', 'YYYY-MM-DD'), 'Shelter A', 'Shelter B', 1);
INSERT INTO animal_transfer_records VALUES (2, TO_DATE('2023-02-15', 'YYYY-MM-DD'), 'Shelter B', 'Shelter C', 2);
INSERT INTO animal_transfer_records VALUES (3, TO_DATE('2023-03-20', 'YYYY-MM-DD'), 'Shelter C', 'Shelter D', 3);
INSERT INTO animal_transfer_records VALUES (4, TO_DATE('2023-04-25', 'YYYY-MM-DD'), 'Shelter D', 'Shelter E', 4);
INSERT INTO animal_transfer_records VALUES (5, TO_DATE('2023-05-30', 'YYYY-MM-DD'), 'Shelter E', 'Shelter F', 5);
INSERT INTO animal_transfer_records VALUES (6, TO_DATE('2023-06-05', 'YYYY-MM-DD'), 'Shelter F', 'Shelter G', 6);
INSERT INTO animal_transfer_records VALUES (7, TO_DATE('2023-07-10', 'YYYY-MM-DD'), 'Shelter G', 'Shelter H', 7);
INSERT INTO animal_transfer_records VALUES (8, TO_DATE('2023-08-15', 'YYYY-MM-DD'), 'Shelter H', 'Shelter I', 8);
INSERT INTO animal_transfer_records VALUES (9, TO_DATE('2023-09-20', 'YYYY-MM-DD'), 'Shelter I', 'Shelter J', 9);
INSERT INTO animal_transfer_records VALUES (10, TO_DATE('2023-10-25', 'YYYY-MM-DD'), 'Shelter J', 'Shelter K', 10);
```

```
Elapsed: 00:00:00.006

1 row inserted.

Elapsed: 00:00:00.007

1 row inserted.

Elapsed: 00:00:00.007
```

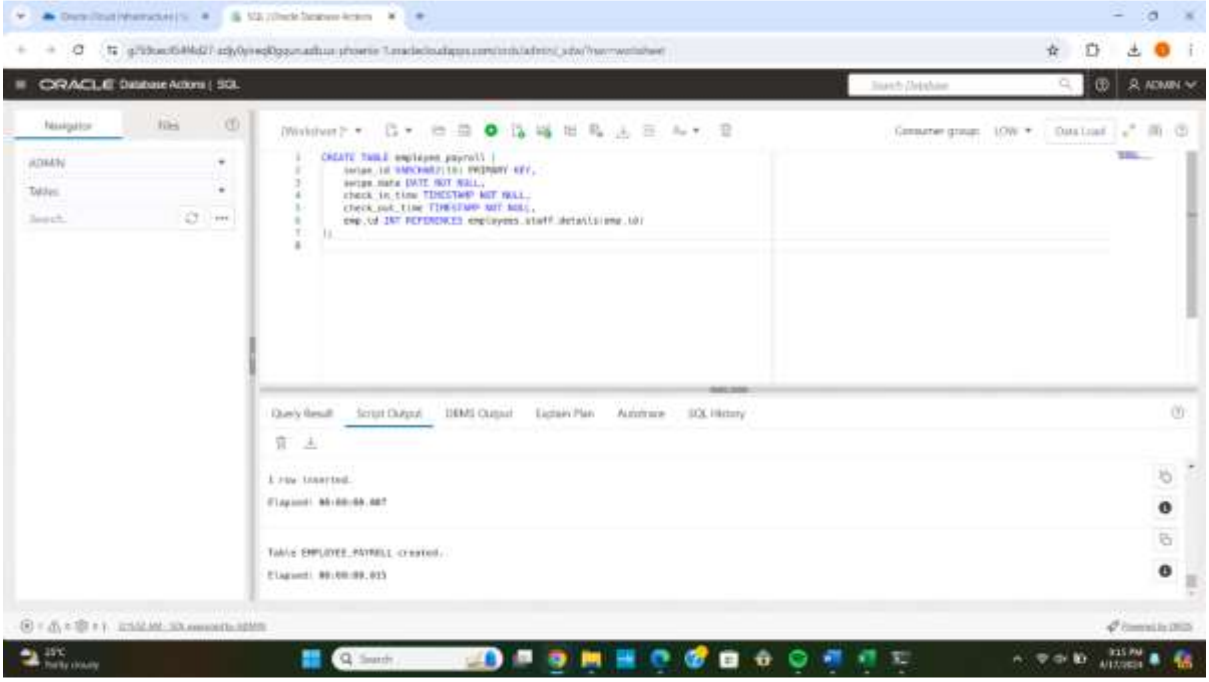


```
Select * from animal_transfer_records;
```

Execution time: 0.016 seconds

	TRANSFER_ID	TRANSFER_DATE	TRANSFER_FROM	TRANSFER_TO	ANIMAL_ID
1	1	1/10/2023, 12:00:00 A	Shelter A	Shelter B	1
2	2	2/15/2023, 12:00:00 A	Shelter B	Shelter C	2
3	3	3/20/2023, 12:00:00 A	Shelter C	Shelter D	3
4	4	4/25/2023, 12:00:00 A	Shelter D	Shelter E	4
5	5	5/30/2023, 12:00:00 A	Shelter E	Shelter F	5
6	6	6/5/2023, 12:00:00 AM	Shelter F	Shelter G	6
7	7	7/10/2023, 12:00:00 A	Shelter G	Shelter H	7
8	8	8/15/2023, 12:00:00 A	Shelter H	Shelter I	8
9	9	9/20/2023, 12:00:00 A	Shelter I	Shelter J	9
10	10	10/25/2023, 12:00:00	Shelter J	Shelter K	10

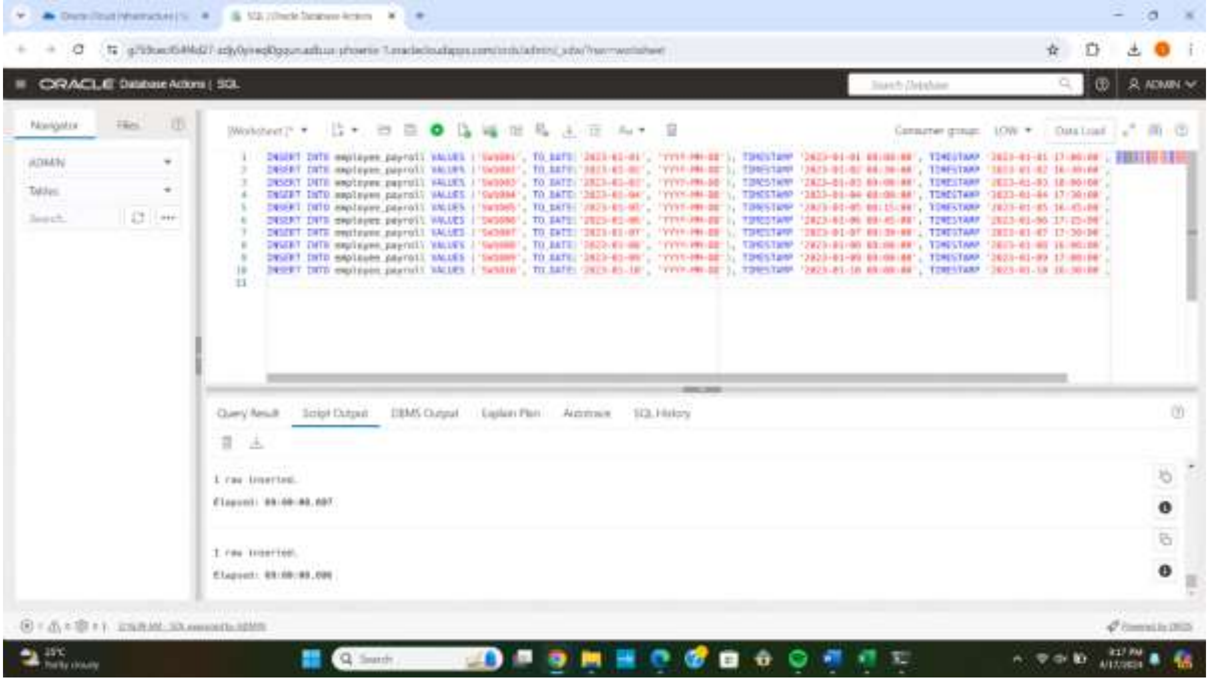
Creating a table named employee_payroll



```
CREATE TABLE employee_payroll (
    swipe_id VARCHAR2(10) PRIMARY KEY,
    swipe_date DATE NOT NULL,
    check_in_time TIMESTAMP NOT NULL,
    check_out_time TIMESTAMP NOT NULL,
    emp_id INT REFERENCES employees.staff_details(emp_id)
);
```

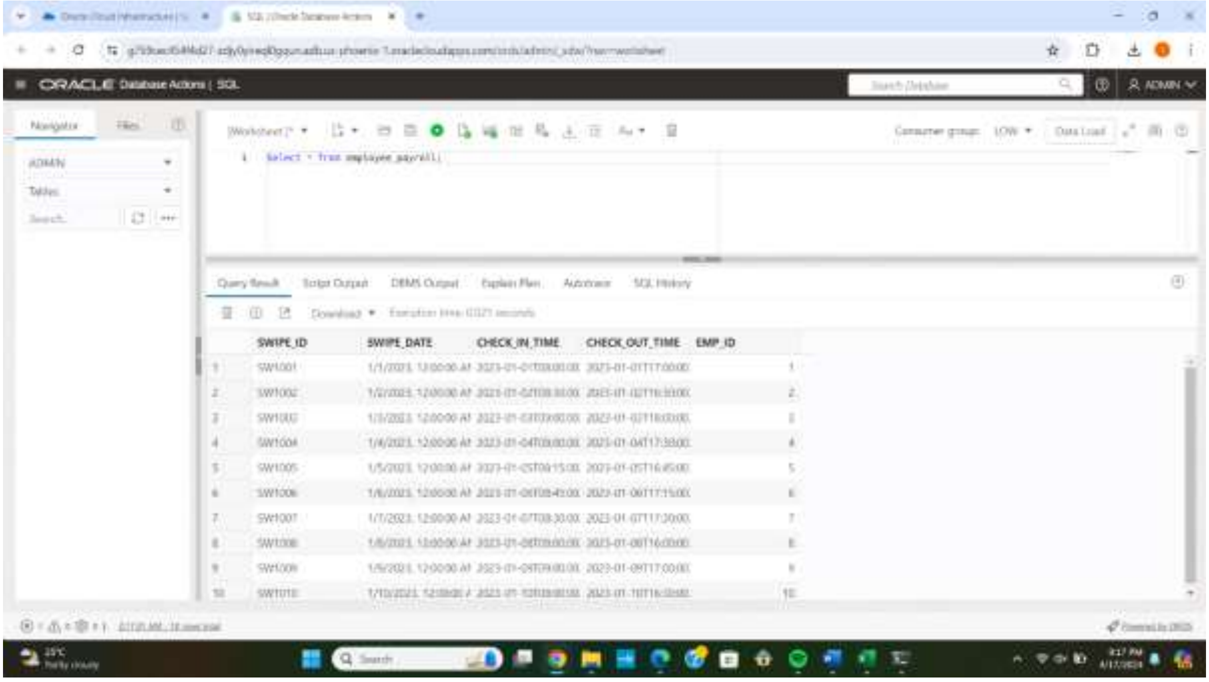
```
1 row inserted.

Elapsed: 00:00:00.007


Table EMPLOYEE_PAYROLL created.

Elapsed: 00:00:00.015
```

Inserting values to employee_payroll



```
INSERT INTO employee_payroll VALUES ('SW0001', TO_DATE('2023-01-01', 'YYYY-MM-DD'), TIMESTAMP '2023-01-01 08:00:00', TIMESTAMP '2023-01-01 17:00:00',
INSERT INTO employee_payroll VALUES ('SW0002', TO_DATE('2023-01-02', 'YYYY-MM-DD'), TIMESTAMP '2023-01-02 08:30:00', TIMESTAMP '2023-01-02 16:30:00',
INSERT INTO employee_payroll VALUES ('SW0003', TO_DATE('2023-01-03', 'YYYY-MM-DD'), TIMESTAMP '2023-01-03 09:00:00', TIMESTAMP '2023-01-03 18:00:00',
INSERT INTO employee_payroll VALUES ('SW0004', TO_DATE('2023-01-04', 'YYYY-MM-DD'), TIMESTAMP '2023-01-04 08:00:00', TIMESTAMP '2023-01-04 17:30:00',
INSERT INTO employee_payroll VALUES ('SW0005', TO_DATE('2023-01-05', 'YYYY-MM-DD'), TIMESTAMP '2023-01-05 08:15:00', TIMESTAMP '2023-01-05 16:45:00',
INSERT INTO employee_payroll VALUES ('SW0006', TO_DATE('2023-01-06', 'YYYY-MM-DD'), TIMESTAMP '2023-01-06 08:45:00', TIMESTAMP '2023-01-06 17:15:00',
INSERT INTO employee_payroll VALUES ('SW0007', TO_DATE('2023-01-07', 'YYYY-MM-DD'), TIMESTAMP '2023-01-07 08:30:00', TIMESTAMP '2023-01-07 17:30:00',
INSERT INTO employee_payroll VALUES ('SW0008', TO_DATE('2023-01-08', 'YYYY-MM-DD'), TIMESTAMP '2023-01-08 08:00:00', TIMESTAMP '2023-01-08 16:00:00',
INSERT INTO employee_payroll VALUES ('SW0009', TO_DATE('2023-01-09', 'YYYY-MM-DD'), TIMESTAMP '2023-01-09 09:00:00', TIMESTAMP '2023-01-09 17:00:00',
INSERT INTO employee_payroll VALUES ('SW0010', TO_DATE('2023-01-10', 'YYYY-MM-DD'), TIMESTAMP '2023-01-10 08:00:00', TIMESTAMP '2023-01-10 16:30:00',
```

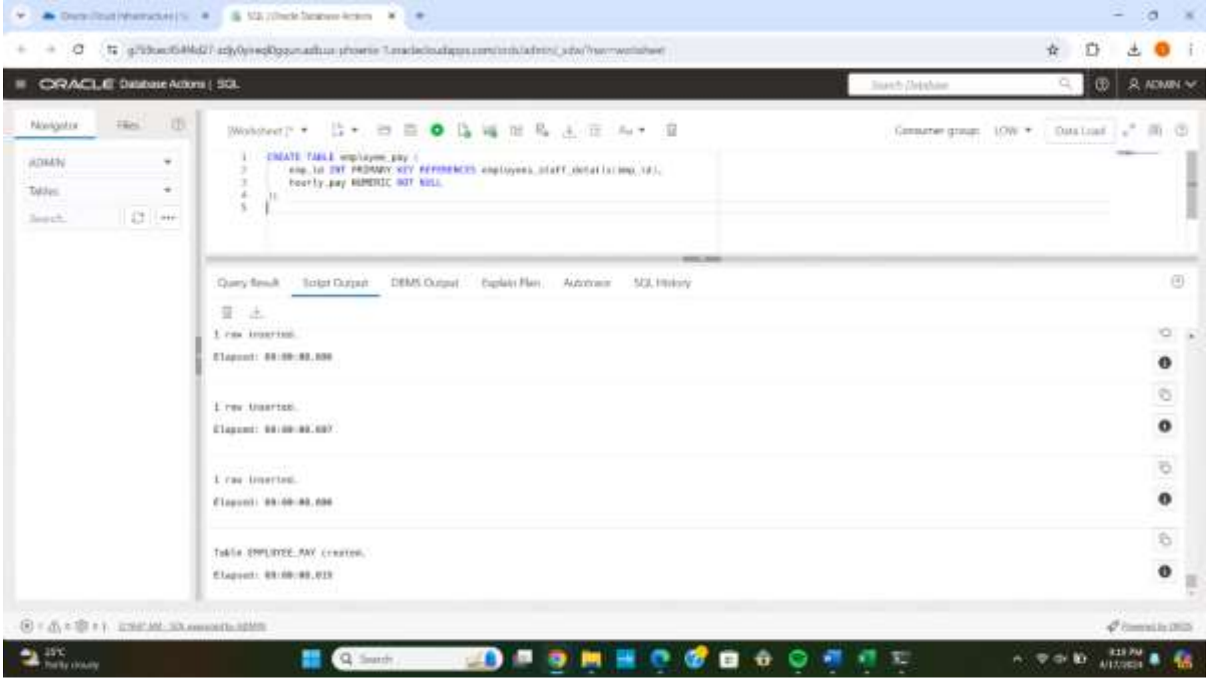
```
1 row inserted.

Elapsed: 00:00:00.007


1 row inserted.

Elapsed: 00:00:00.006
```



```
select * from employee_pay/all;
```

	SWIPE_ID	SWIPE_DATE	CHECK_IN_TIME	CHECK_OUT_TIME	EMP_ID
1	SW1001	1/1/2023, 12:00:00 AI	2023-01-01T09:00:00	2023-01-01T17:00:00	1
2	SW1002	1/2/2023, 12:00:00 AI	2023-01-02T08:30:00	2023-01-02T16:30:00	2
3	SW1003	1/3/2023, 12:00:00 AI	2023-01-03T09:00:00	2023-01-03T18:00:00	3
4	SW1004	1/4/2023, 12:00:00 AI	2023-01-04T09:00:00	2023-01-04T17:30:00	4
5	SW1005	1/5/2023, 12:00:00 AI	2023-01-05T09:15:00	2023-01-05T16:45:00	5
6	SW1006	1/6/2023, 12:00:00 AI	2023-01-06T08:45:00	2023-01-06T17:15:00	6
7	SW1007	1/7/2023, 12:00:00 AI	2023-01-07T08:30:00	2023-01-07T17:30:00	7
8	SW1008	1/8/2023, 12:00:00 AI	2023-01-08T09:00:00	2023-01-08T16:00:00	8
9	SW1009	1/9/2023, 12:00:00 AI	2023-01-09T09:00:00	2023-01-09T17:00:00	9
10	SW1010	1/10/2023, 12:00:00 /	2023-01-10T08:00:00	2023-01-10T16:30:00	10

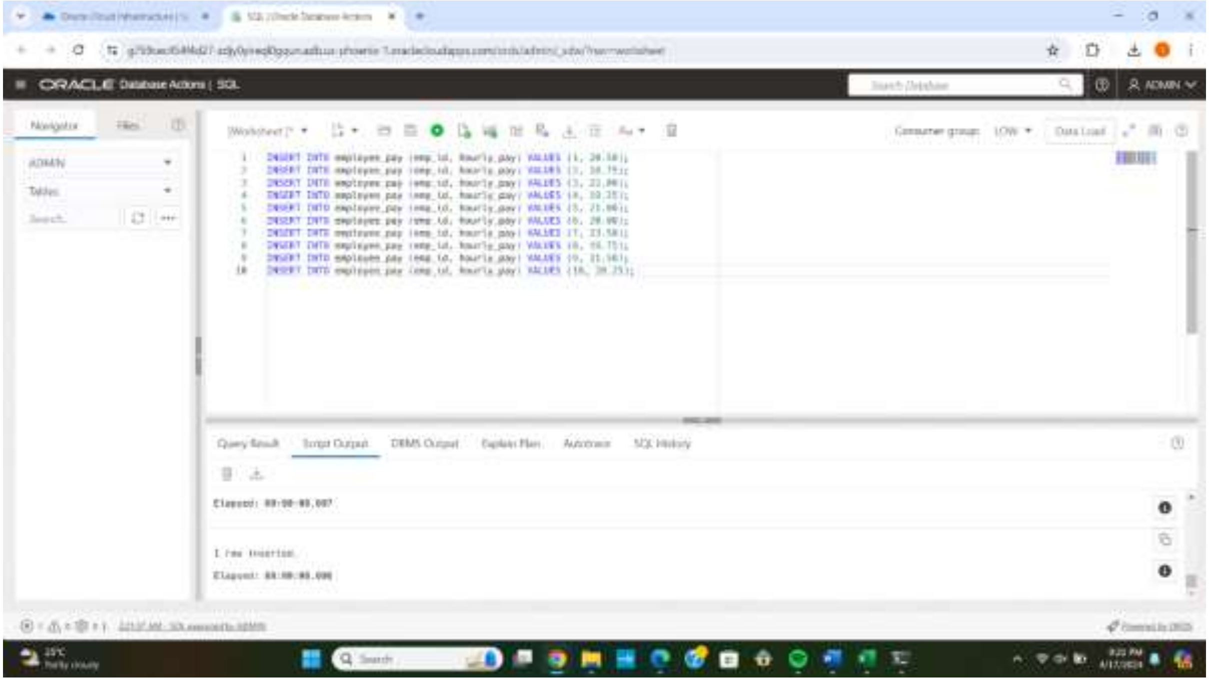
Creating a table named employee_pay(New)



```
CREATE TABLE employee_pay (
    emp_id INT PRIMARY KEY REFERENCES employees_staff_details(emp_id),
    hourly_pay NUMERIC NOT NULL
);
```

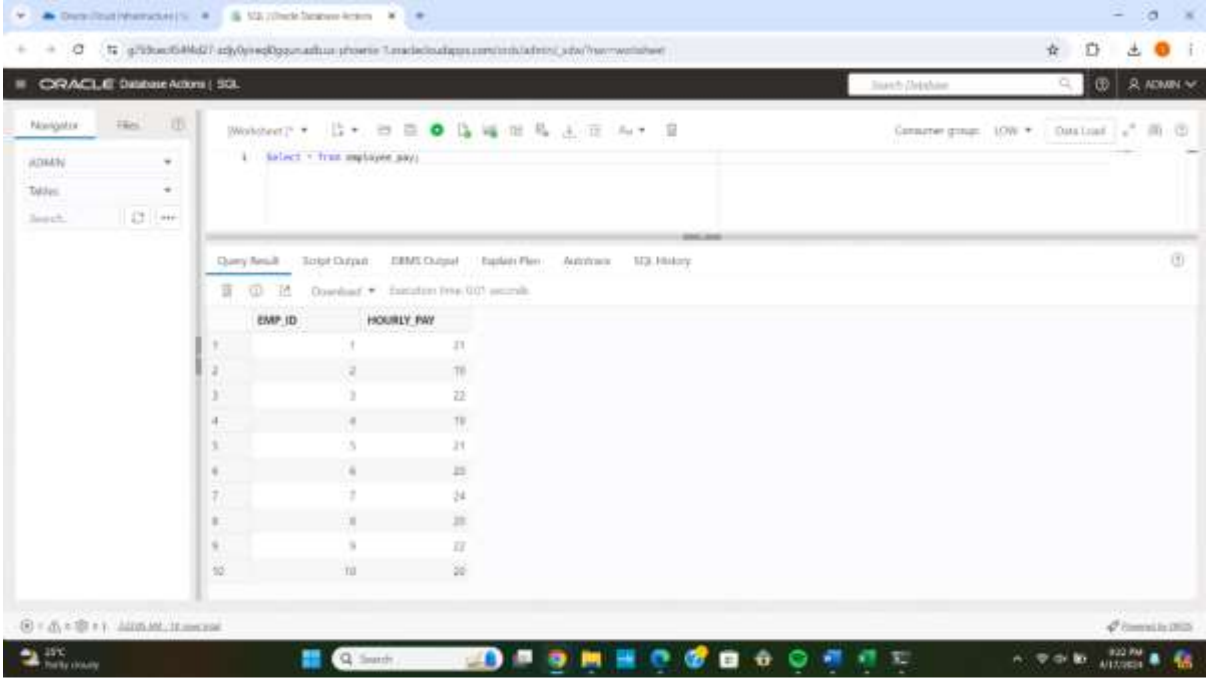
```
1 row inserted.

Elapsed: 00:00:00.000


1 row inserted.

Elapsed: 00:00:00.007


1 row inserted.

Elapsed: 00:00:00.006


Table EMPLOYEE_PAY created.

Elapsed: 00:00:00.019
```

Inserting values to employee_pay(New)



```sql
INSERT INTO employee_pay (emp_id, hourly_pay) VALUES (1, 20.50);
INSERT INTO employee_pay (emp_id, hourly_pay) VALUES (2, 18.75);
INSERT INTO employee_pay (emp_id, hourly_pay) VALUES (3, 22.00);
INSERT INTO employee_pay (emp_id, hourly_pay) VALUES (4, 19.25);
INSERT INTO employee_pay (emp_id, hourly_pay) VALUES (5, 21.00);
INSERT INTO employee_pay (emp_id, hourly_pay) VALUES (6, 20.00);
INSERT INTO employee_pay (emp_id, hourly_pay) VALUES (7, 23.50);
INSERT INTO employee_pay (emp_id, hourly_pay) VALUES (8, 19.75);
INSERT INTO employee_pay (emp_id, hourly_pay) VALUES (9, 21.50);
INSERT INTO employee_pay (emp_id, hourly_pay) VALUES (10, 20.25);
```

```
Elapsed: 00:00:00.007


1 row inserted.

Elapsed: 00:00:00.006
```

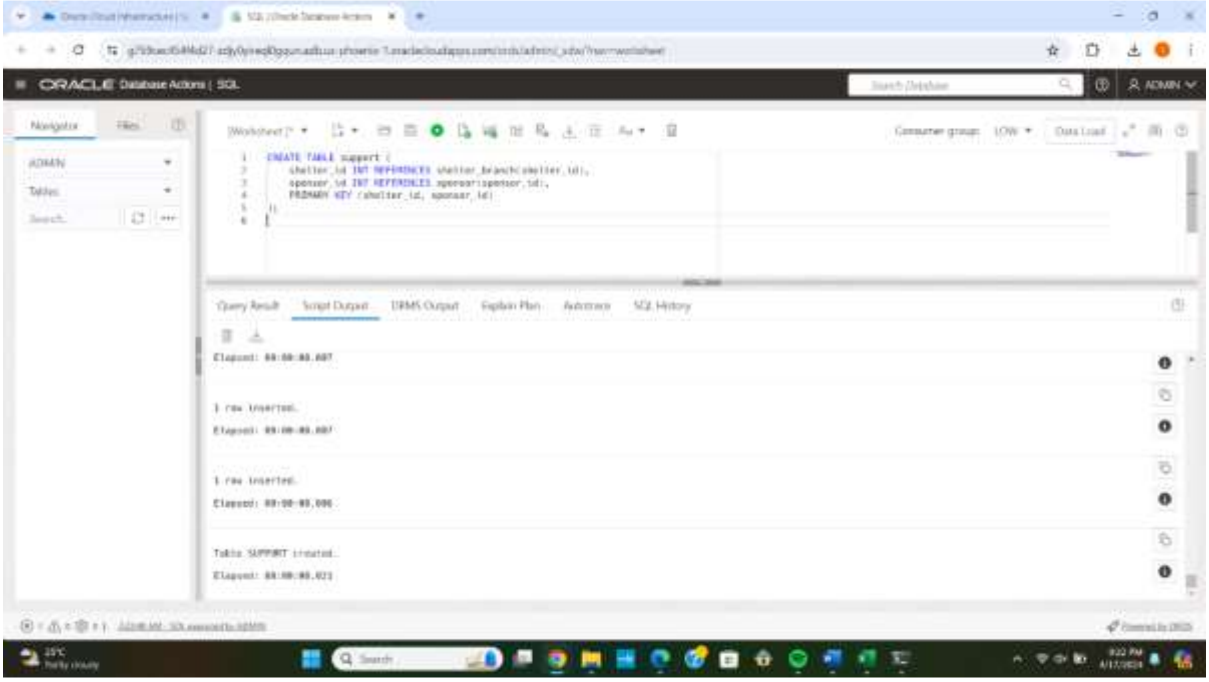


```sql
select * from employee_pay;
```

Execution time: 0.01 seconds

	EMP_ID	HOURLY_PAY
1	1	21
2	2	19
3	3	22
4	4	19
5	5	21
6	6	20
7	7	24
8	8	20
9	9	22
10	10	20

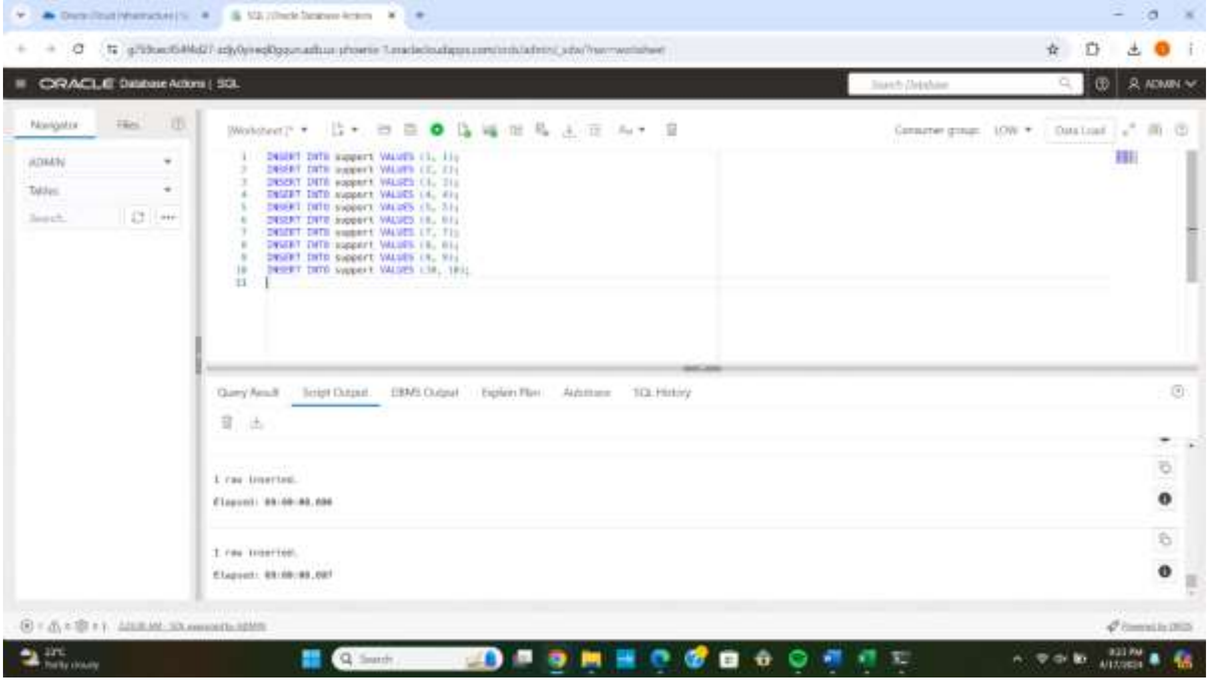
Creating a table named support



```
CREATE TABLE support (
    shelter_id INT REFERENCES shelter_branch(shelter_id),
    sponsor_id INT REFERENCES sponsor(sponsor_id),
    PRIMARY KEY (shelter_id, sponsor_id)
);
```

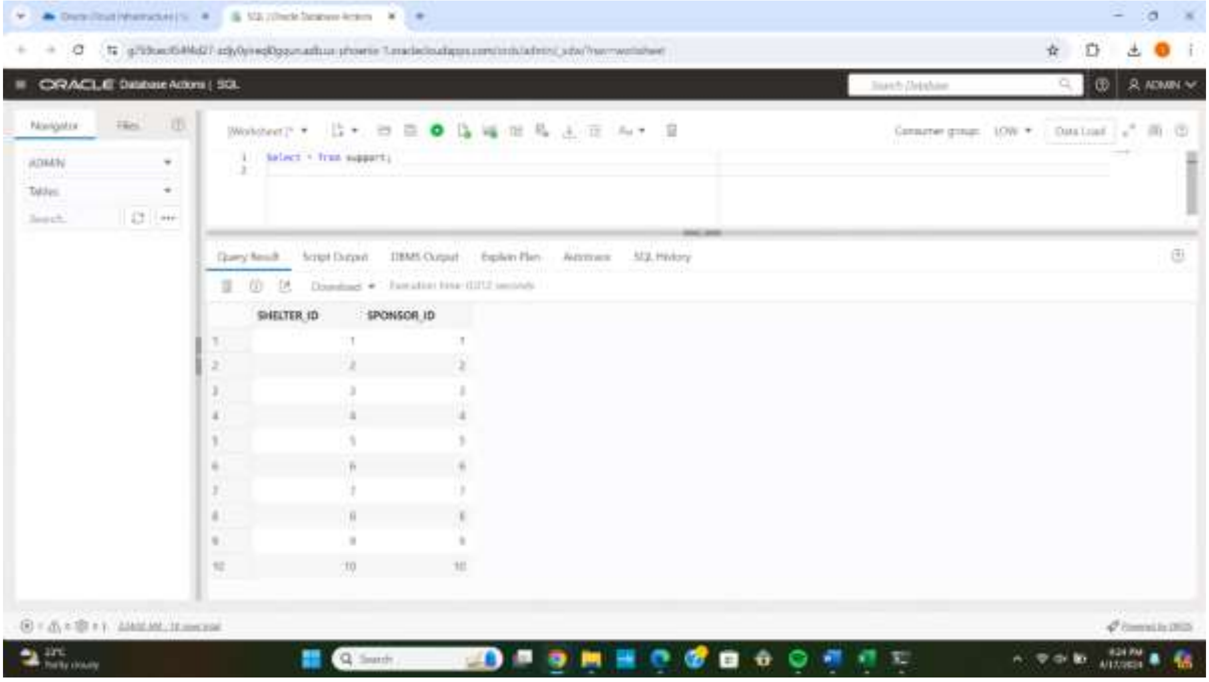
```
Elapsed: 00:00:00.007


1 row inserted.

Elapsed: 00:00:00.007


1 row inserted.

Elapsed: 00:00:00.006


Table SUPPORT created.

Elapsed: 00:00:00.021
```

Inserting values to support



```
INSERT INTO support VALUES (1, 1);
INSERT INTO support VALUES (2, 2);
INSERT INTO support VALUES (3, 3);
INSERT INTO support VALUES (4, 4);
INSERT INTO support VALUES (5, 5);
INSERT INTO support VALUES (6, 6);
INSERT INTO support VALUES (7, 7);
INSERT INTO support VALUES (8, 8);
INSERT INTO support VALUES (9, 9);
INSERT INTO support VALUES (10, 10);
```

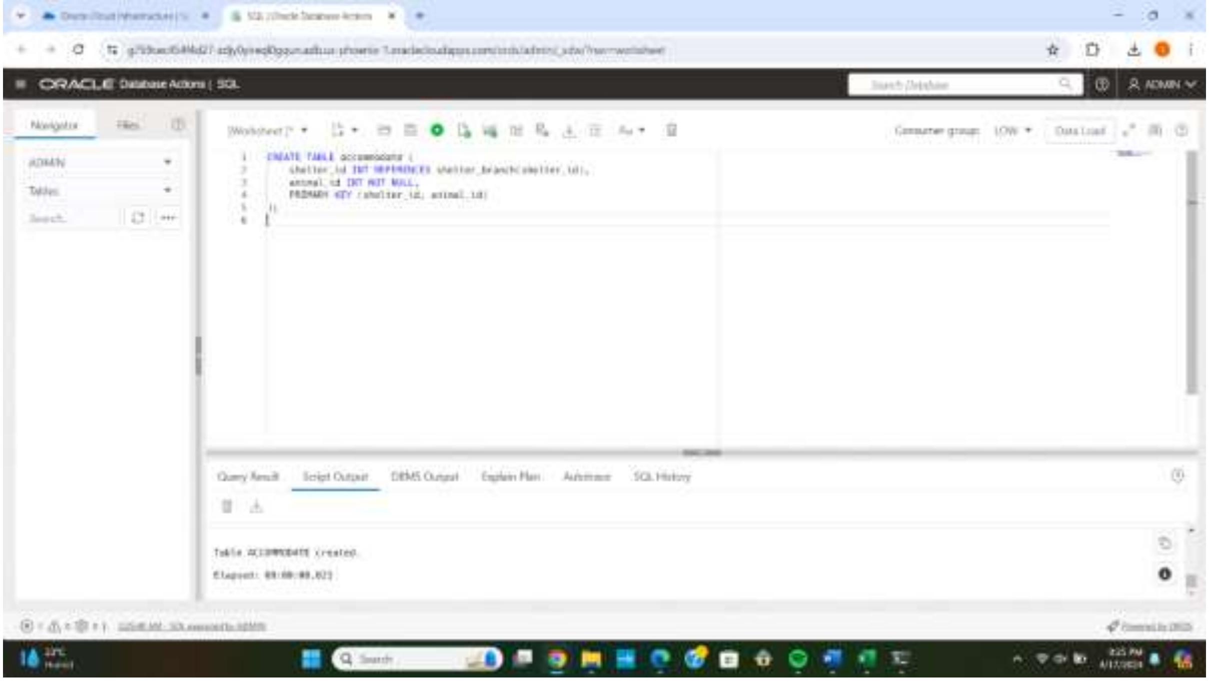
```
1 row inserted.

Elapsed: 00:00:00.006


1 row inserted.

Elapsed: 00:00:00.007
```



```
select * from support;
```

	SHELTER_ID	SPONSOR_ID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10

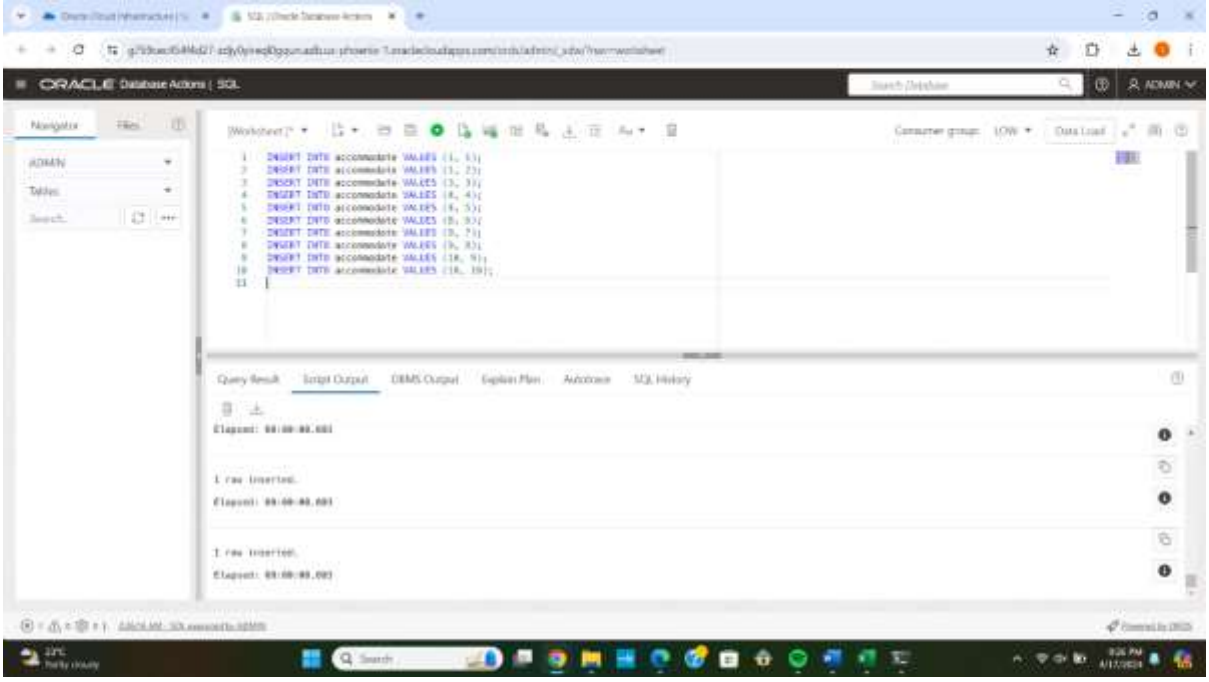
Creating a table named accommodate



```
CREATE TABLE accommodate (
    shelter_id INT REFERENCES shelter_branch(shelter_id),
    animal_id INT NOT NULL,
    PRIMARY KEY (shelter_id, animal_id)
);
```

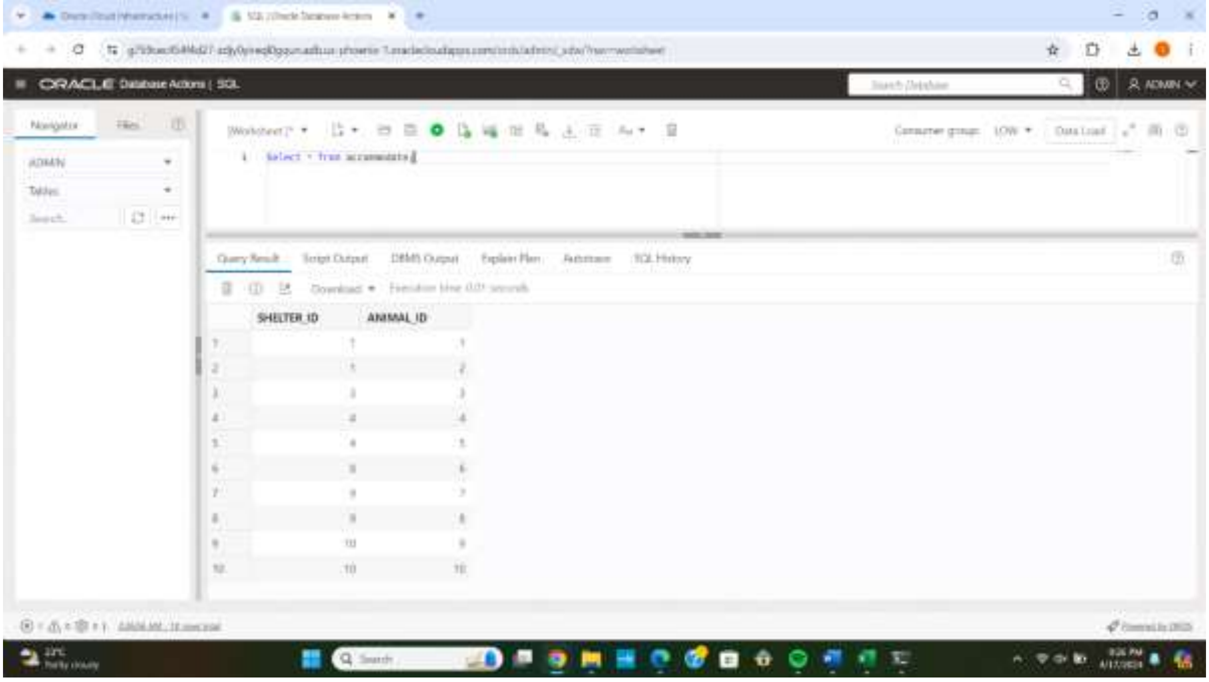
```
Table ACCOMMODATE created.

Elapsed: 00:00:00.023
```

Inserting values to accommodate



```
INSERT INTO accommodate VALUES (1, 1);
INSERT INTO accommodate VALUES (1, 2);
INSERT INTO accommodate VALUES (3, 3);
INSERT INTO accommodate VALUES (4, 4);
INSERT INTO accommodate VALUES (4, 5);
INSERT INTO accommodate VALUES (8, 6);
INSERT INTO accommodate VALUES (9, 7);
INSERT INTO accommodate VALUES (9, 8);
INSERT INTO accommodate VALUES (10, 9);
INSERT INTO accommodate VALUES (10, 10);
```

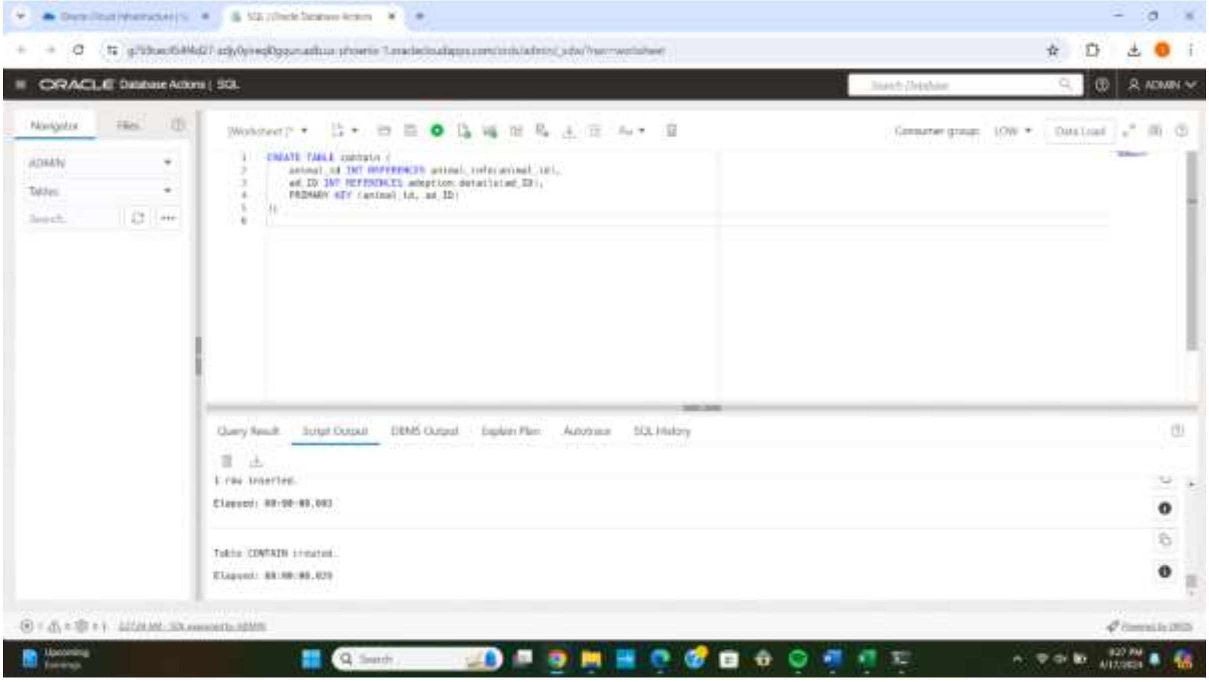
```
Elapsed: 00:00:00.001


1 row inserted.

Elapsed: 00:00:00.001


1 row inserted.

Elapsed: 00:00:00.003
```



```
select * from accommodate
```

	SHELTER_ID	ANIMAL_ID
1	1	1
2	1	2
3	3	3
4	4	4
5	4	5
6	8	6
7	9	7
8	9	8
9	10	9
10	10	10

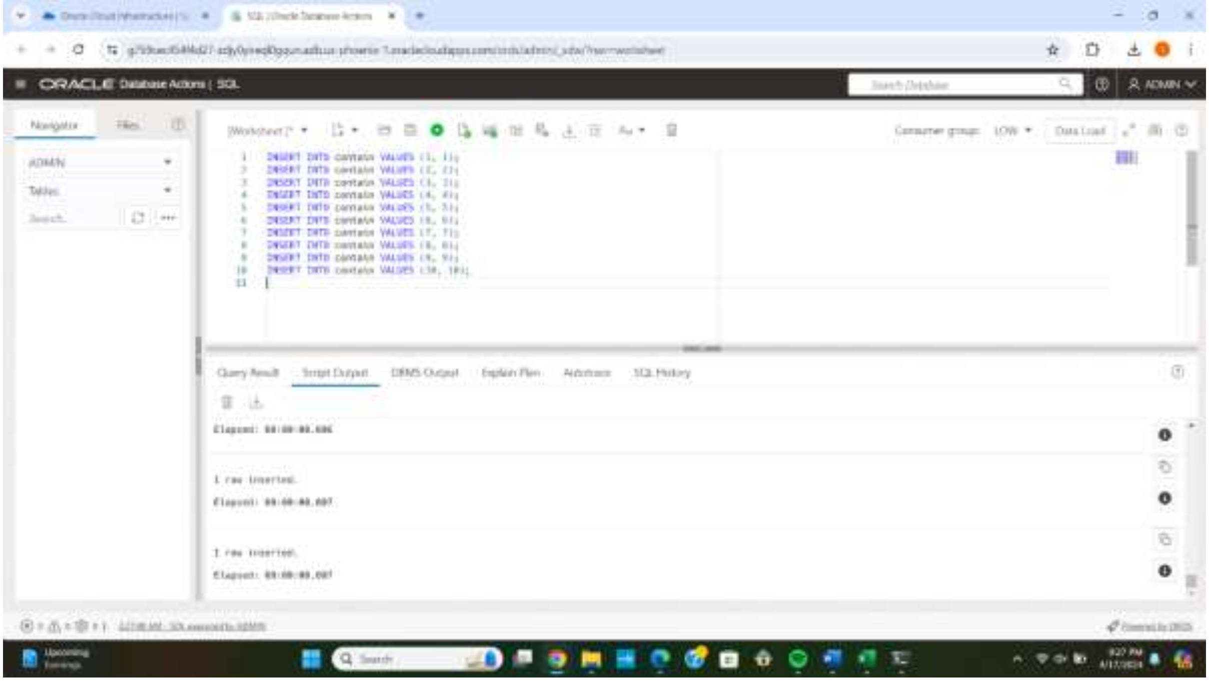
Creating a table named contain



```
CREATE TABLE contain (
    animal_id INT REFERENCES animal_info(animal_id),
    ad_ID INT REFERENCES adoption_details(ad_ID),
    PRIMARY KEY (animal_id, ad_ID)
);
```

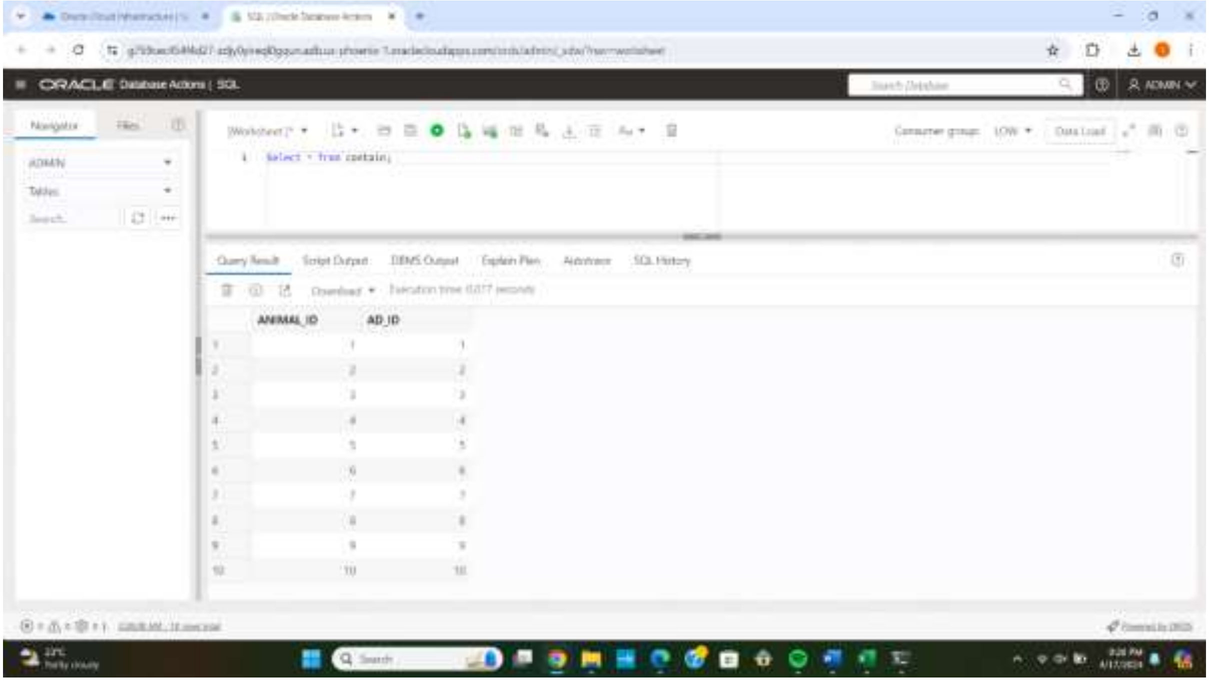
```
1 row inserted.

Elapsed: 00:00:00.003

Table CONTAIN created.

Elapsed: 00:00:00.029
```

Inserting values to contain



```
INSERT INTO contain VALUES (1, 1);
INSERT INTO contain VALUES (2, 2);
INSERT INTO contain VALUES (3, 3);
INSERT INTO contain VALUES (4, 4);
INSERT INTO contain VALUES (5, 5);
INSERT INTO contain VALUES (6, 6);
INSERT INTO contain VALUES (7, 7);
INSERT INTO contain VALUES (8, 8);
INSERT INTO contain VALUES (9, 9);
INSERT INTO contain VALUES (10, 10);
```

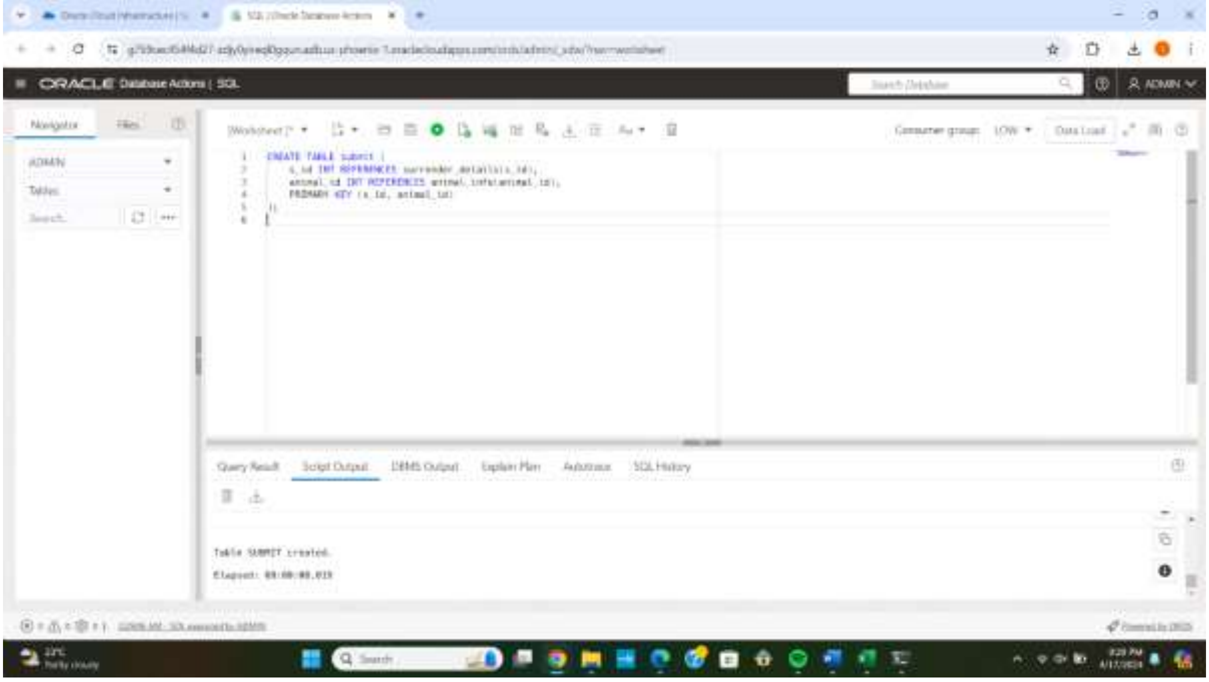
```
Elapsed: 00:00:00.006

1 row inserted.

Elapsed: 00:00:00.007

1 row inserted.

Elapsed: 00:00:00.007
```



```
select * from contain;
```

	ANIMAL_ID	AD_ID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10

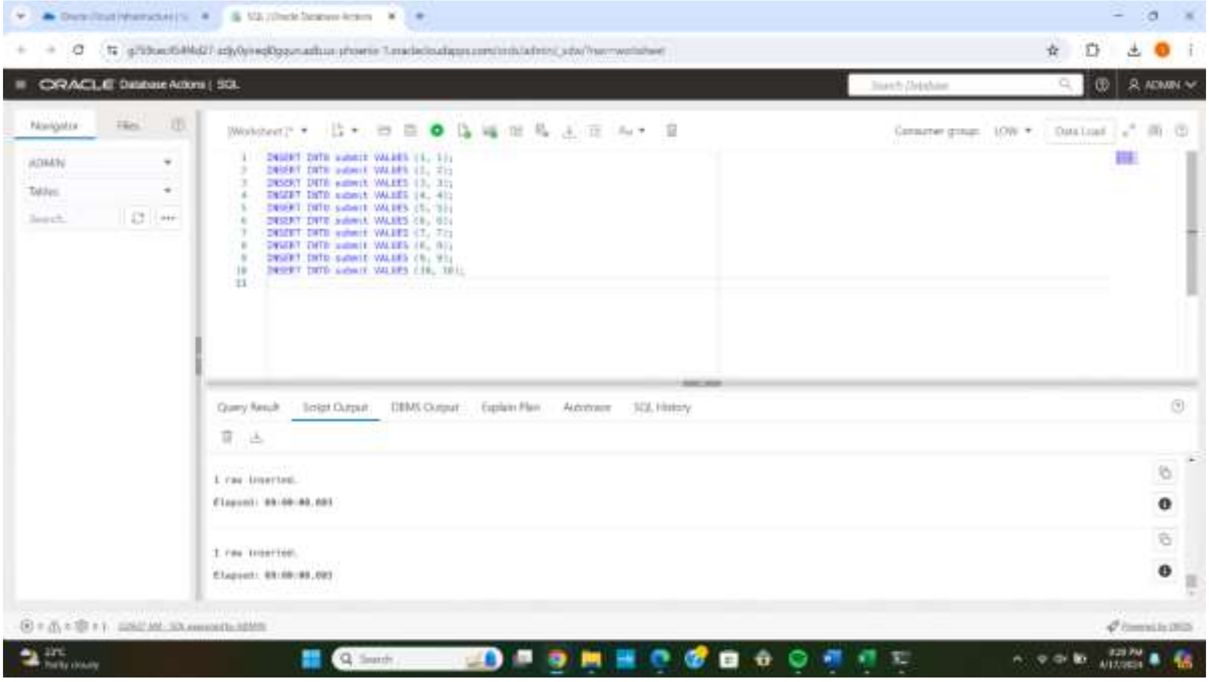
Creating a table named submit



```
CREATE TABLE submit (
    s_id INT REFERENCES surrender_details(s_id),
    animal_id INT REFERENCES animal_info(animal_id),
    PRIMARY KEY (s_id, animal_id)
);
```

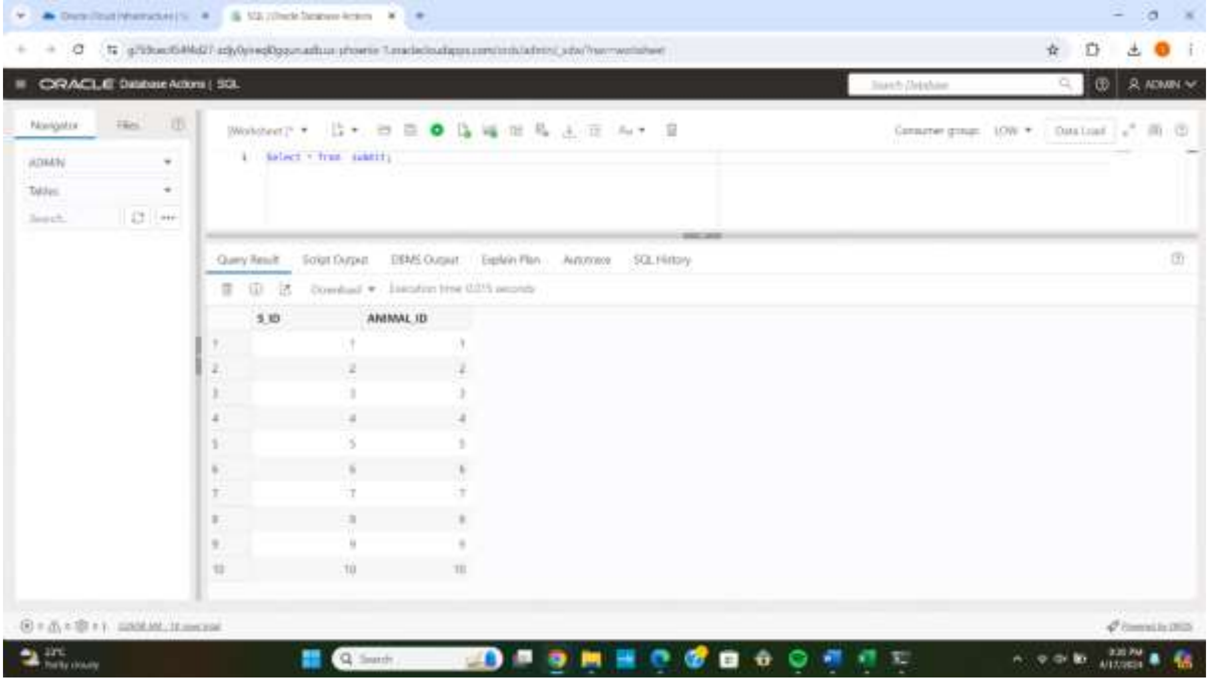
```
Table SUBMIT created.

Elapsed: 00:00:00.019
```

Inserting values to submit



```
INSERT INTO submit VALUES (1, 1);
INSERT INTO submit VALUES (2, 2);
INSERT INTO submit VALUES (3, 3);
INSERT INTO submit VALUES (4, 4);
INSERT INTO submit VALUES (5, 5);
INSERT INTO submit VALUES (6, 6);
INSERT INTO submit VALUES (7, 7);
INSERT INTO submit VALUES (8, 8);
INSERT INTO submit VALUES (9, 9);
INSERT INTO submit VALUES (10, 10);
```

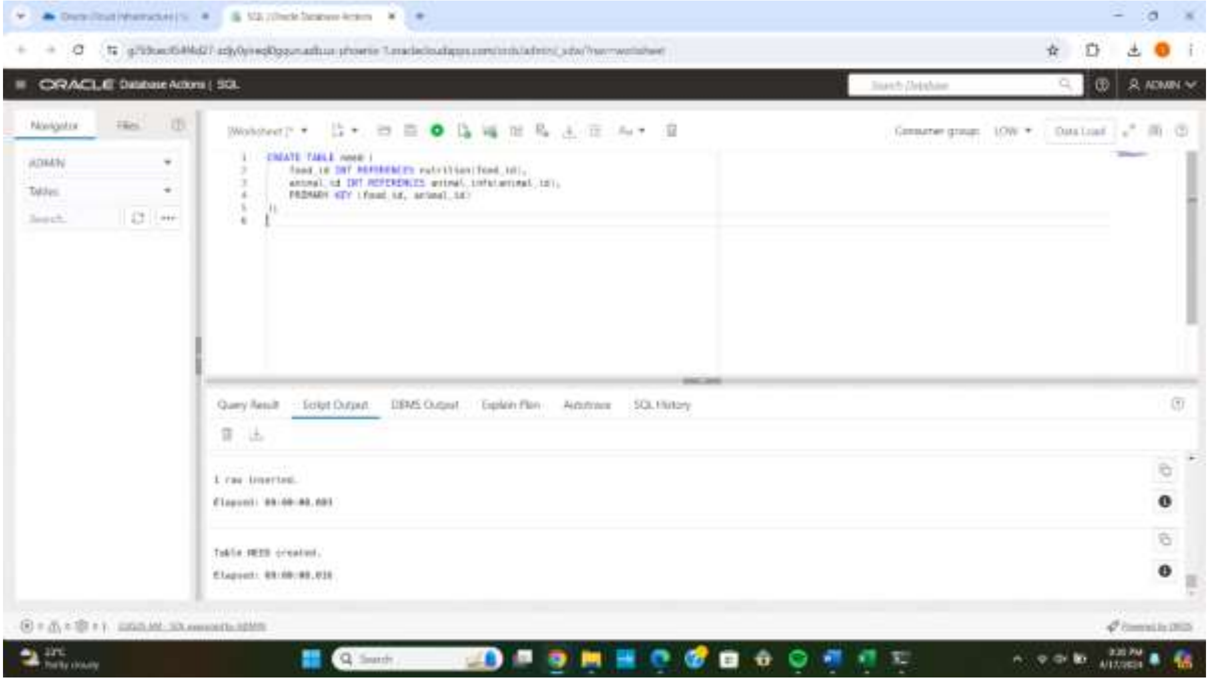
```
1 row inserted.

Elapsed: 00:00:00.003


1 row inserted.

Elapsed: 00:00:00.003
```



```
Select * from submit;
```

	S_ID	ANIMAL_ID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10

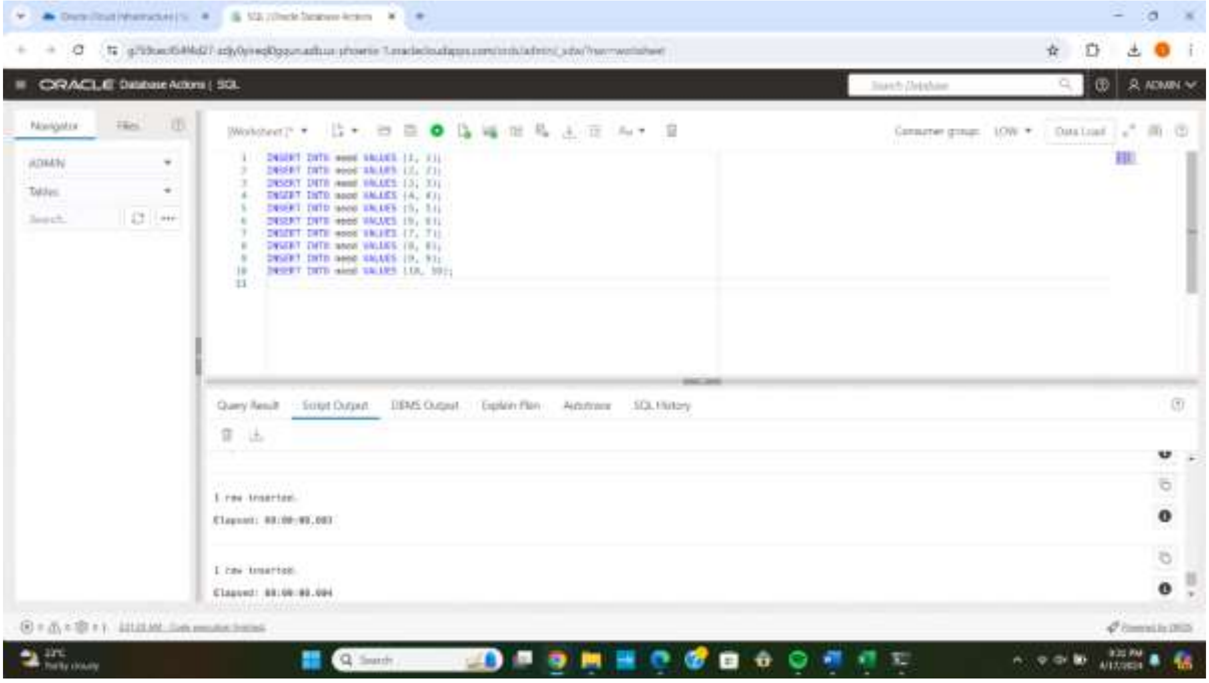
Creating a table named need



```
CREATE TABLE need (
    food_id INT REFERENCES nutrition(food_id),
    animal_id INT REFERENCES animal_info(animal_id),
    PRIMARY KEY (food_id, animal_id)
);
```

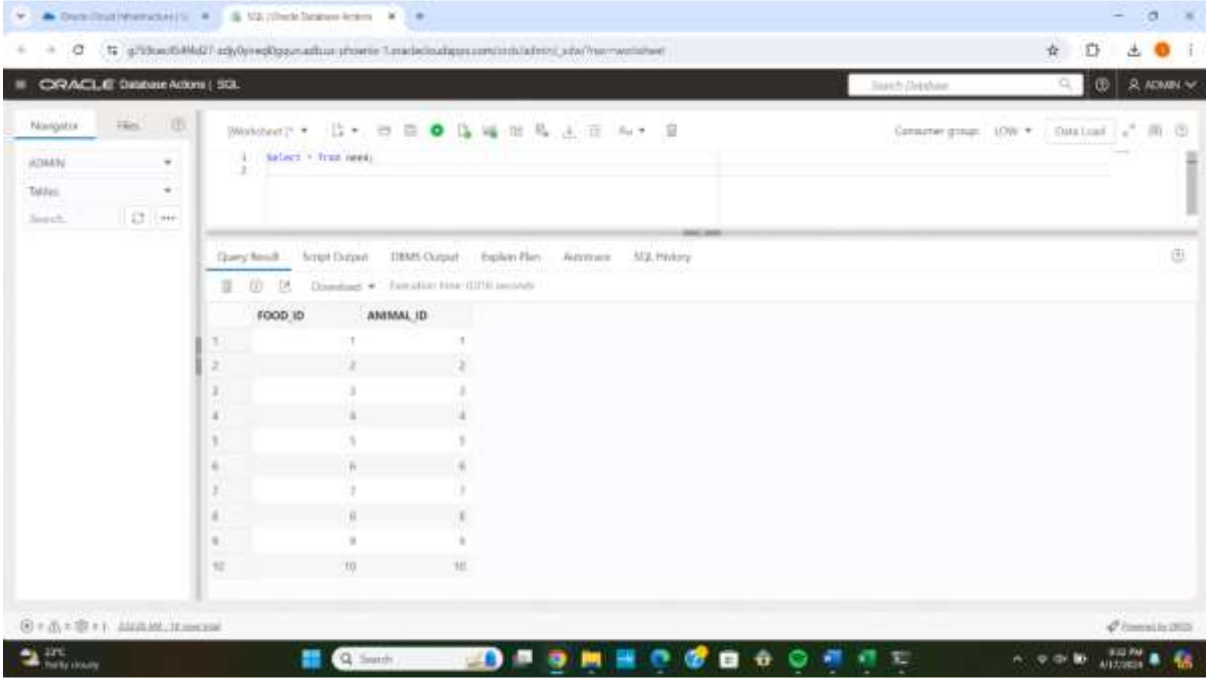
```
1 row inserted.

Elapsed: 00:00:00.003

Table NEED created.

Elapsed: 00:00:00.018
```

Inserting values to need



```
INSERT INTO need VALUES (1, 1);
INSERT INTO need VALUES (2, 2);
INSERT INTO need VALUES (3, 3);
INSERT INTO need VALUES (4, 4);
INSERT INTO need VALUES (5, 5);
INSERT INTO need VALUES (6, 6);
INSERT INTO need VALUES (7, 7);
INSERT INTO need VALUES (8, 8);
INSERT INTO need VALUES (9, 9);
INSERT INTO need VALUES (10, 10);
```

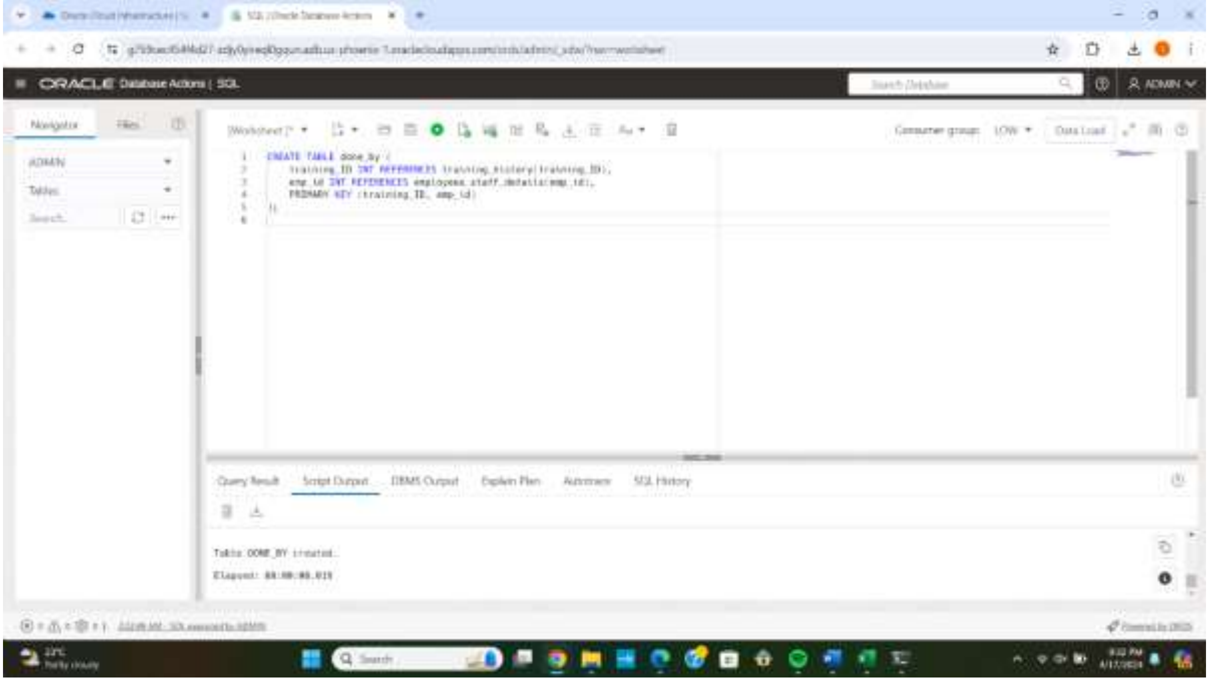
```
1 row inserted.

Elapsed: 00:00:00.003

1 row inserted.

Elapsed: 00:00:00.004
```



```
select * from need;
```

	FOOD_ID	ANIMAL_ID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10

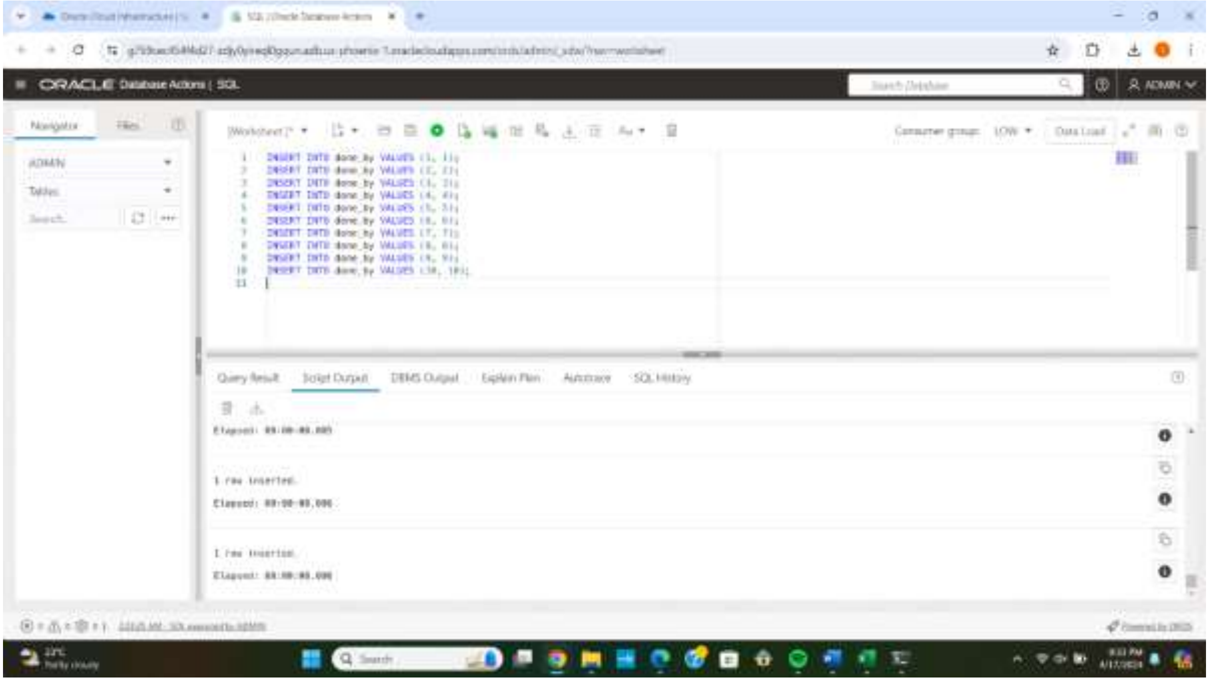
Creating a table named done_by



```
CREATE TABLE done_by (
    training_ID INT REFERENCES training_history(training_ID),
    emp_id INT REFERENCES employees_staff_details(emp_id),
    PRIMARY KEY (training_ID, emp_id)
);
```

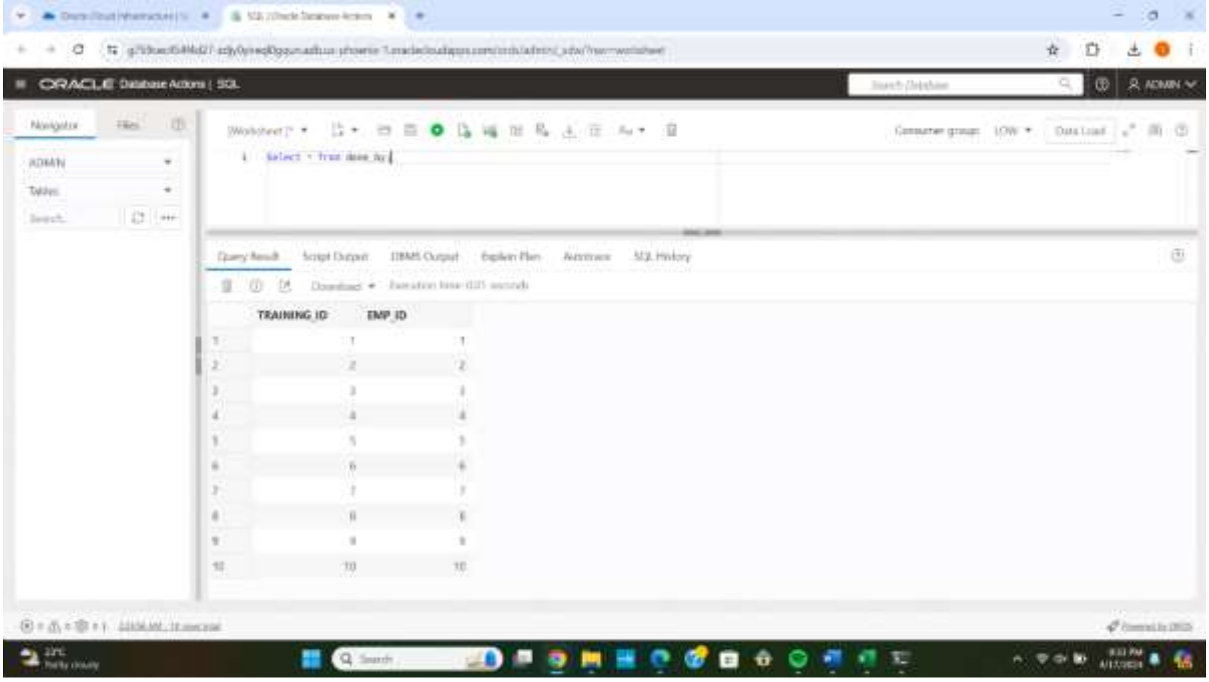
```
Table DONE_BY created.

Elapsed: 00:00:00.019
```

Inserting values to done_by



```
INSERT INTO done_by VALUES (1, 1);
INSERT INTO done_by VALUES (2, 2);
INSERT INTO done_by VALUES (3, 3);
INSERT INTO done_by VALUES (4, 4);
INSERT INTO done_by VALUES (5, 5);
INSERT INTO done_by VALUES (6, 6);
INSERT INTO done_by VALUES (7, 7);
INSERT INTO done_by VALUES (8, 8);
INSERT INTO done_by VALUES (9, 9);
INSERT INTO done_by VALUES (10, 10);
```

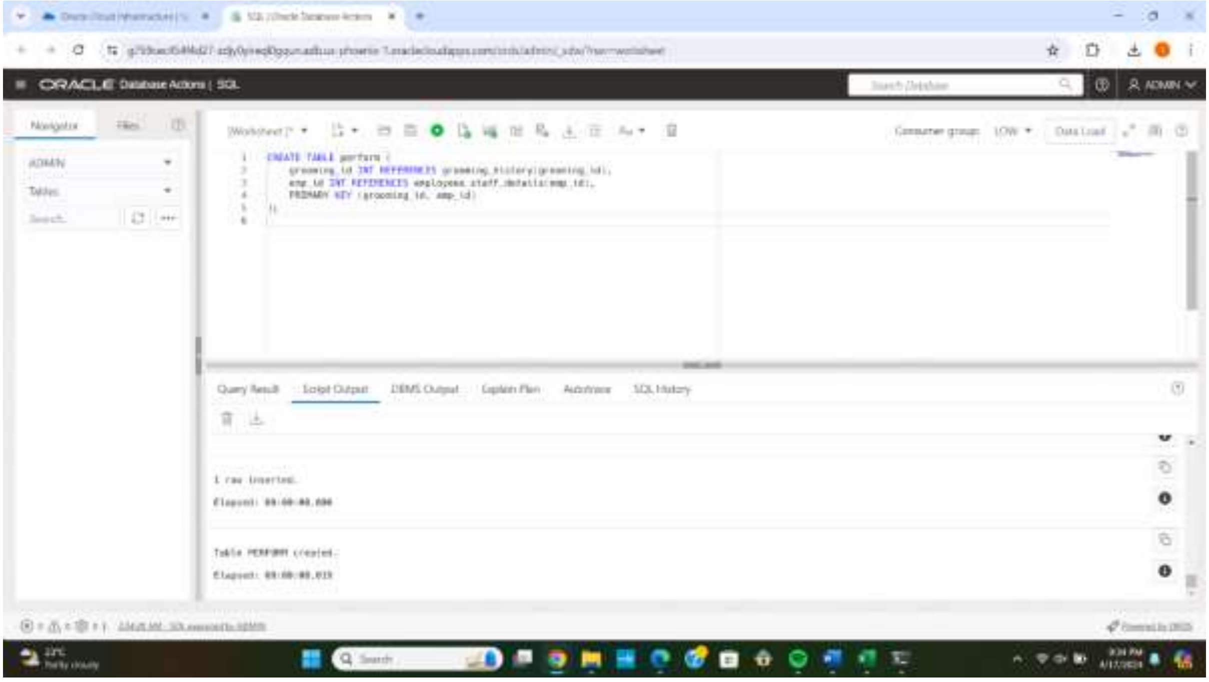
```
Elapsed: 00:00:00.005


1 row inserted.

Elapsed: 00:00:00.006


1 row inserted.

Elapsed: 00:00:00.006
```



```
select * from done_by;
```

	TRAINING_ID	EMP_ID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10

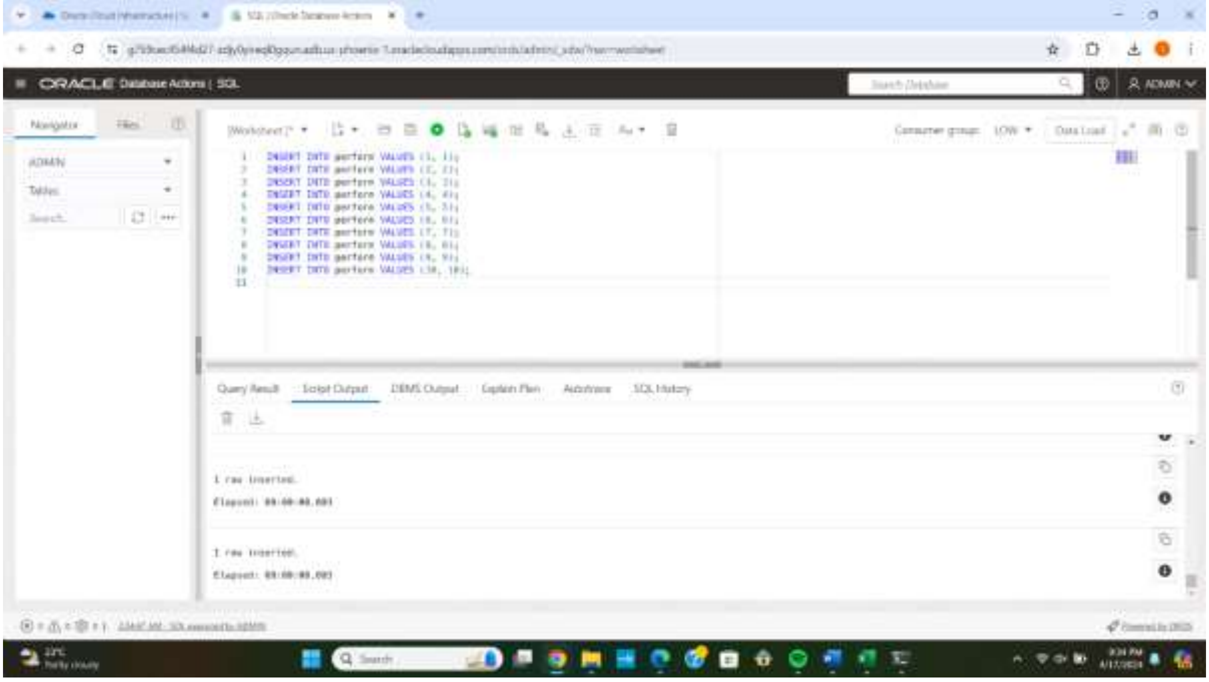
Creating a table named perform



```
CREATE TABLE perform (
    grooming_id INT REFERENCES grooming_history(grooming_id),
    emp_id INT REFERENCES employees_staff_details(emp_id),
    PRIMARY KEY (grooming_id, emp_id)
);
```

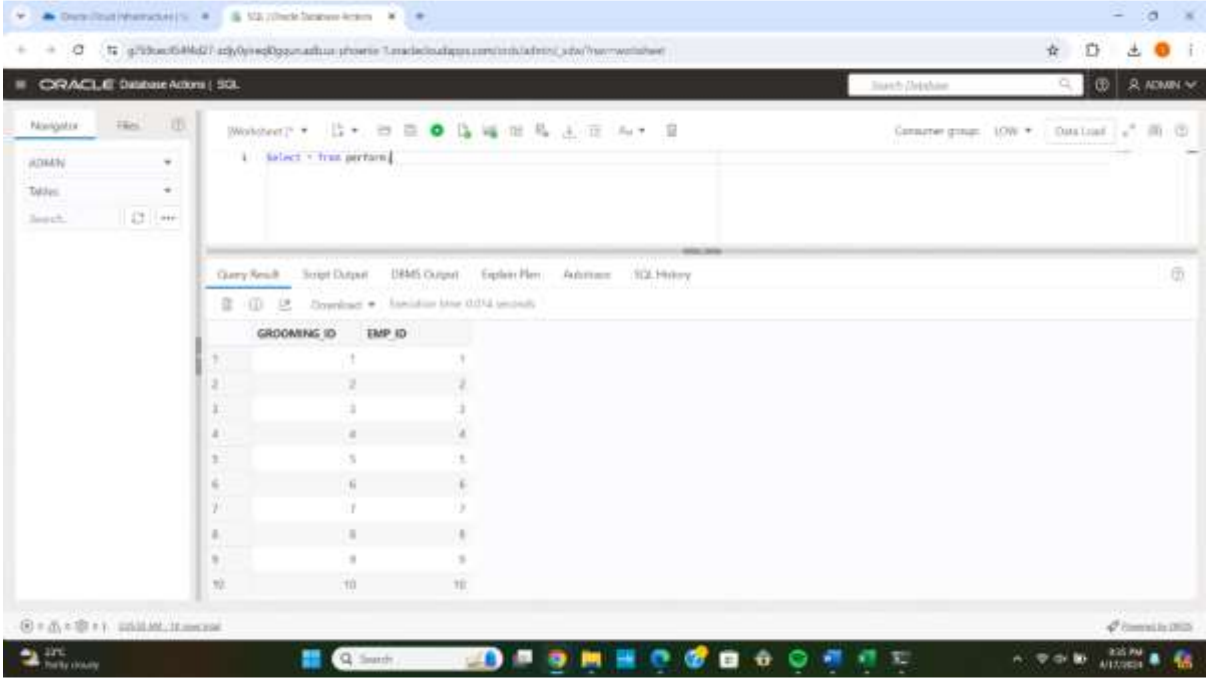
```
1 row inserted.

Elapsed: 00:00:00.006

Table PERFORM created.

Elapsed: 00:00:00.019
```

Inserting values to perform



```
INSERT INTO perform VALUES (1, 1);
INSERT INTO perform VALUES (2, 2);
INSERT INTO perform VALUES (3, 3);
INSERT INTO perform VALUES (4, 4);
INSERT INTO perform VALUES (5, 5);
INSERT INTO perform VALUES (6, 6);
INSERT INTO perform VALUES (7, 7);
INSERT INTO perform VALUES (8, 8);
INSERT INTO perform VALUES (9, 9);
INSERT INTO perform VALUES (10, 10);
```

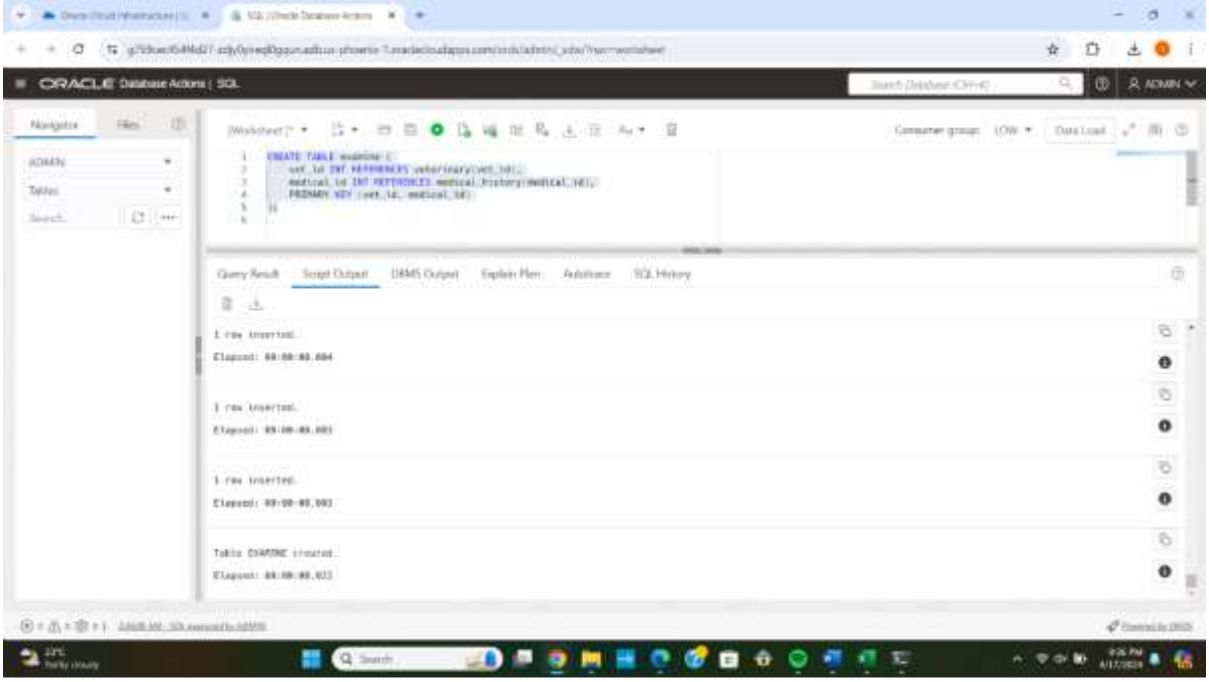
```
1 row inserted.

Elapsed: 00:00:00.003

1 row inserted.

Elapsed: 00:00:00.003
```



```
Select * from perform;
```

	GROOMING_ID	EMP_ID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10

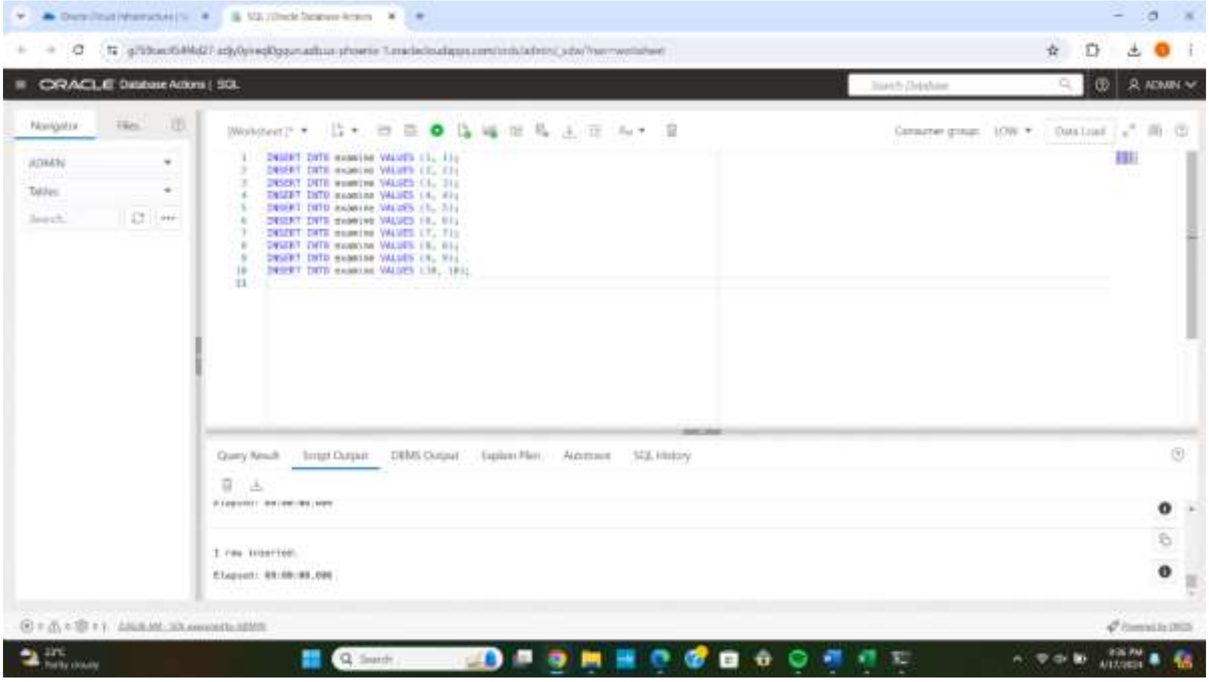
Creating a table named examine



```
CREATE TABLE examine (
    vet_id INT REFERENCES veterinary(vet_id),
    medical_id INT REFERENCES medical_history(medical_id),
    PRIMARY KEY (vet_id, medical_id)
);
```

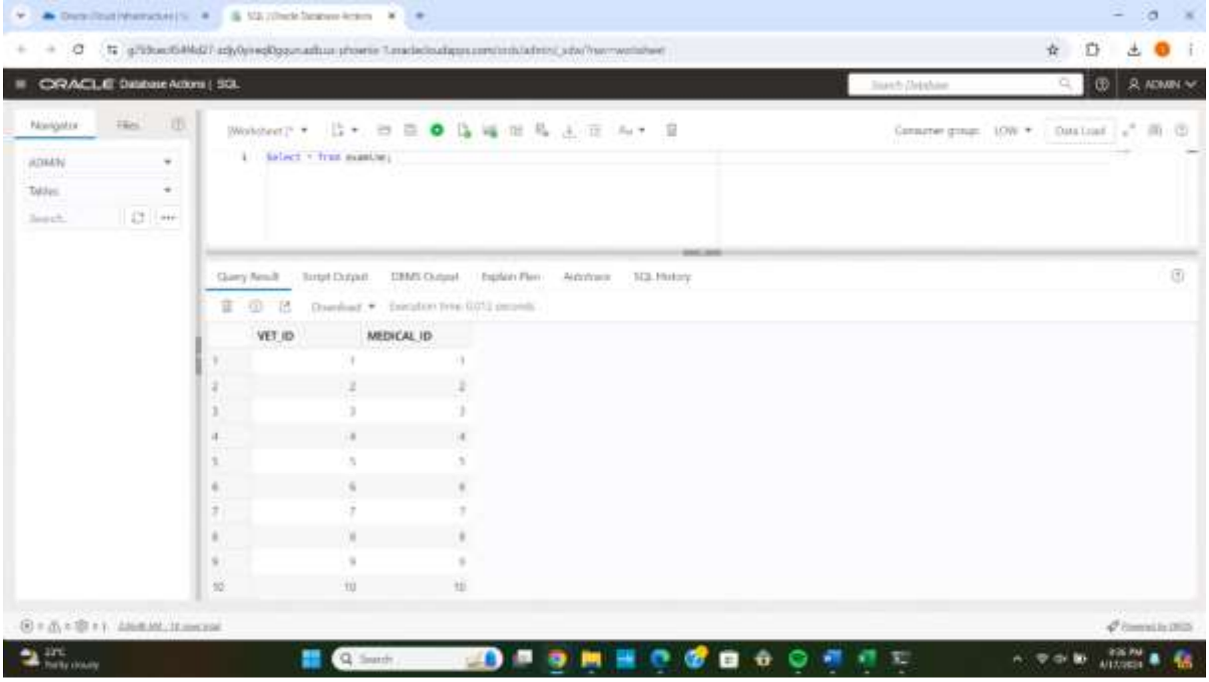
```
1 row inserted.

Elapsed: 00:00:00.004

1 row inserted.

Elapsed: 00:00:00.003

1 row inserted.

Elapsed: 00:00:00.003

Table EXAMINE created.

Elapsed: 00:00:00.023
```

Inserting values to examine



```sql
INSERT INTO examine VALUES (1, 1);
INSERT INTO examine VALUES (2, 2);
INSERT INTO examine VALUES (3, 3);
INSERT INTO examine VALUES (4, 4);
INSERT INTO examine VALUES (5, 5);
INSERT INTO examine VALUES (6, 6);
INSERT INTO examine VALUES (7, 7);
INSERT INTO examine VALUES (8, 8);
INSERT INTO examine VALUES (9, 9);
INSERT INTO examine VALUES (10, 10);
```

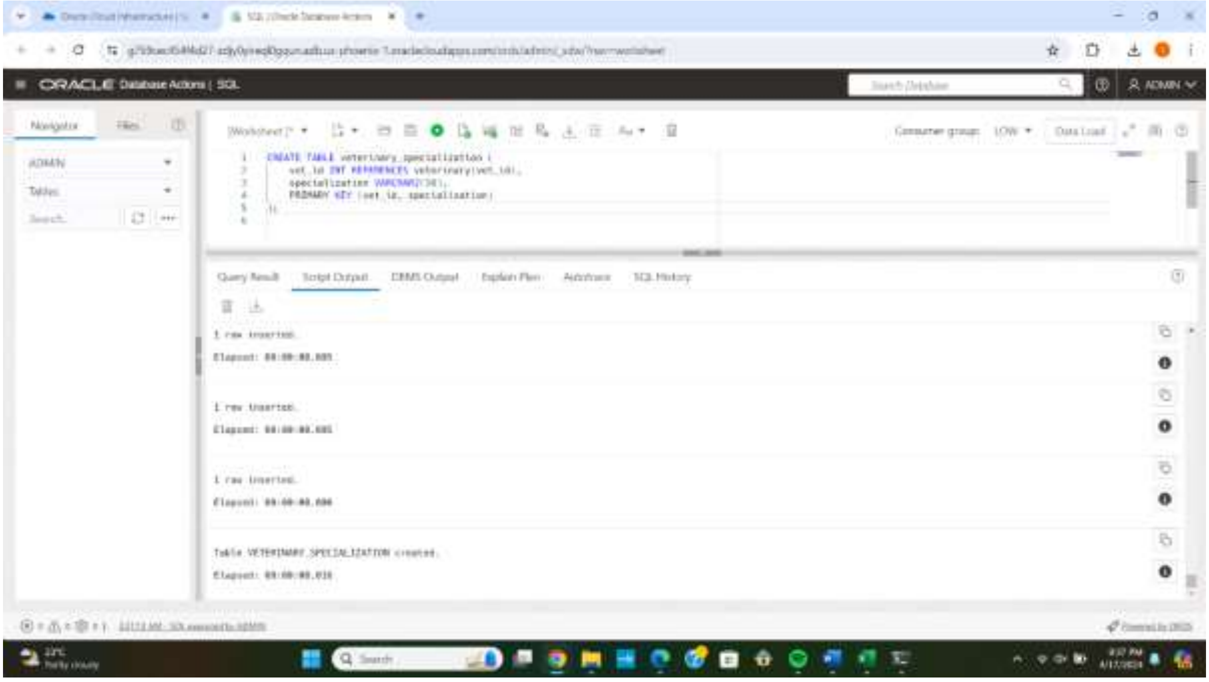
```
Elapsed: 00:00:00.006


1 row inserted.

Elapsed: 00:00:00.006
```



```sql
Select * from examine;
```

	VET_ID	MEDICAL_ID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10

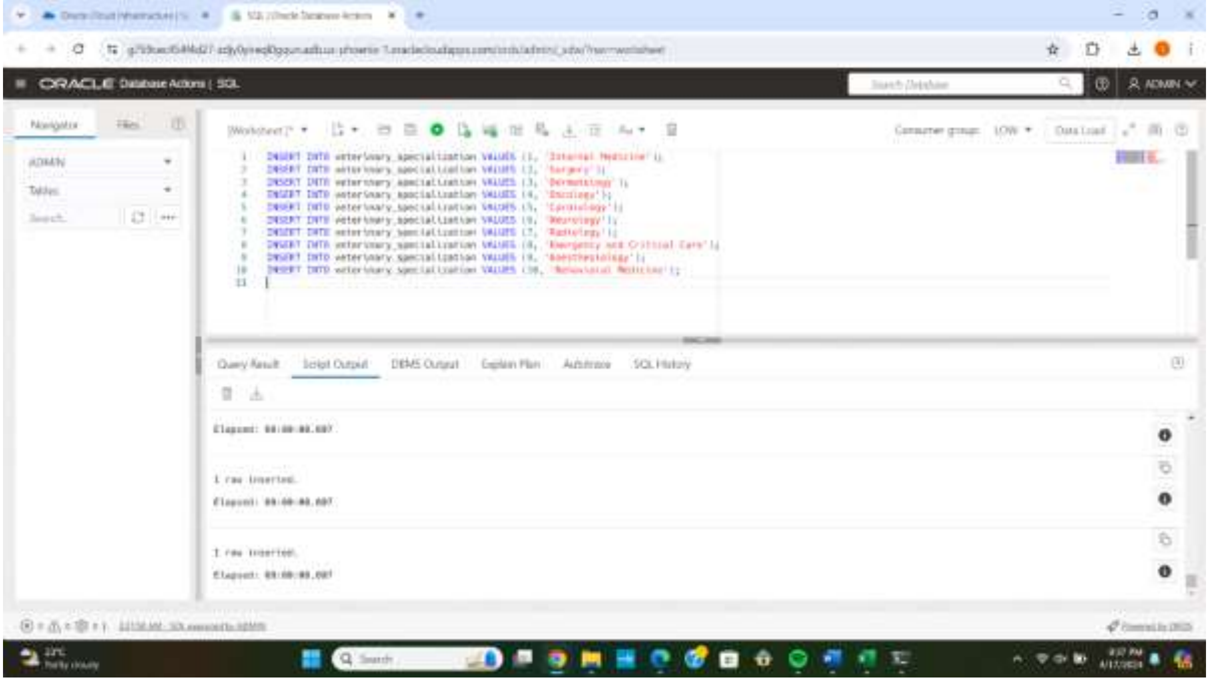
Creating a table named veterinary_specialization



```
CREATE TABLE veterinary_specialization (
    vet_id INT REFERENCES veterinary(vet_id),
    specialization VARCHAR2(30),
    PRIMARY KEY (vet_id, specialization)
);
```

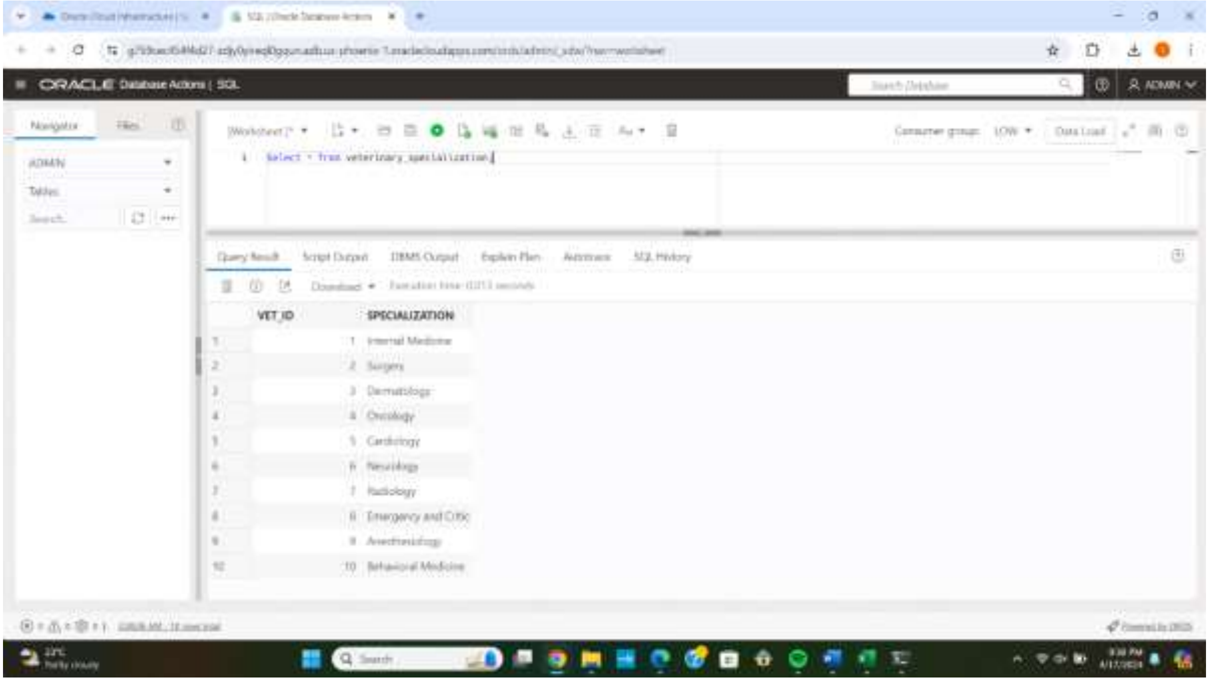
```
1 row inserted.

Elapsed: 00:00:00.005

1 row inserted.

Elapsed: 00:00:00.005

1 row inserted.

Elapsed: 00:00:00.006

Table VETERINARY_SPECIALIZATION created.

Elapsed: 00:00:00.016
```

Inserting values to veterinary_specialization



```
INSERT INTO veterinary_specialization VALUES (1, 'Internal Medicine');
INSERT INTO veterinary_specialization VALUES (2, 'Surgery');
INSERT INTO veterinary_specialization VALUES (3, 'Dermatology');
INSERT INTO veterinary_specialization VALUES (4, 'Oncology');
INSERT INTO veterinary_specialization VALUES (5, 'Cardiology');
INSERT INTO veterinary_specialization VALUES (6, 'Neurology');
INSERT INTO veterinary_specialization VALUES (7, 'Radiology');
INSERT INTO veterinary_specialization VALUES (8, 'Emergency and Critical Care');
INSERT INTO veterinary_specialization VALUES (9, 'Anesthesiology');
INSERT INTO veterinary_specialization VALUES (10, 'Behavioral Medicine');
```

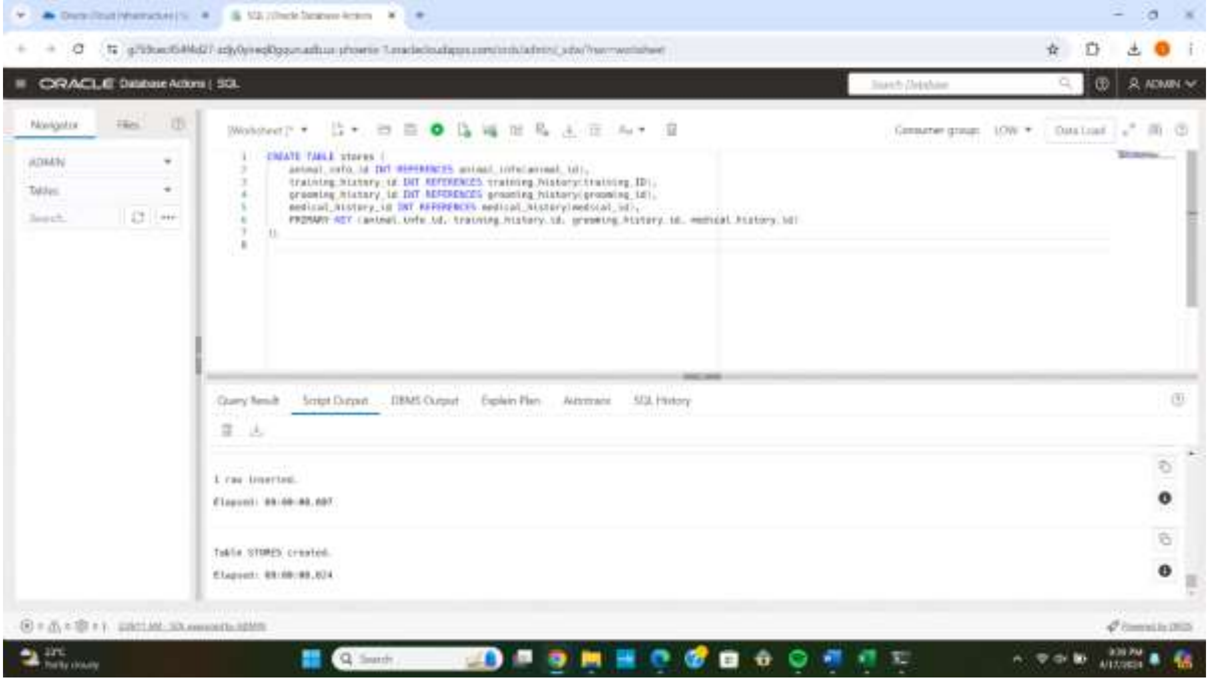
```
Elapsed: 00:00:00.007

1 row inserted.

Elapsed: 00:00:00.007

1 row inserted.

Elapsed: 00:00:00.007
```



```
select * from veterinary_specialization;
```

	VET_ID	SPECIALIZATION
1	1	Internal Medicine
2	2	Surgery
3	3	Dermatology
4	4	Oncology
5	5	Cardiology
6	6	Neurology
7	7	Radiology
8	8	Emergency and Critic
9	9	Anesthesiology
10	10	Behavioral Medicine

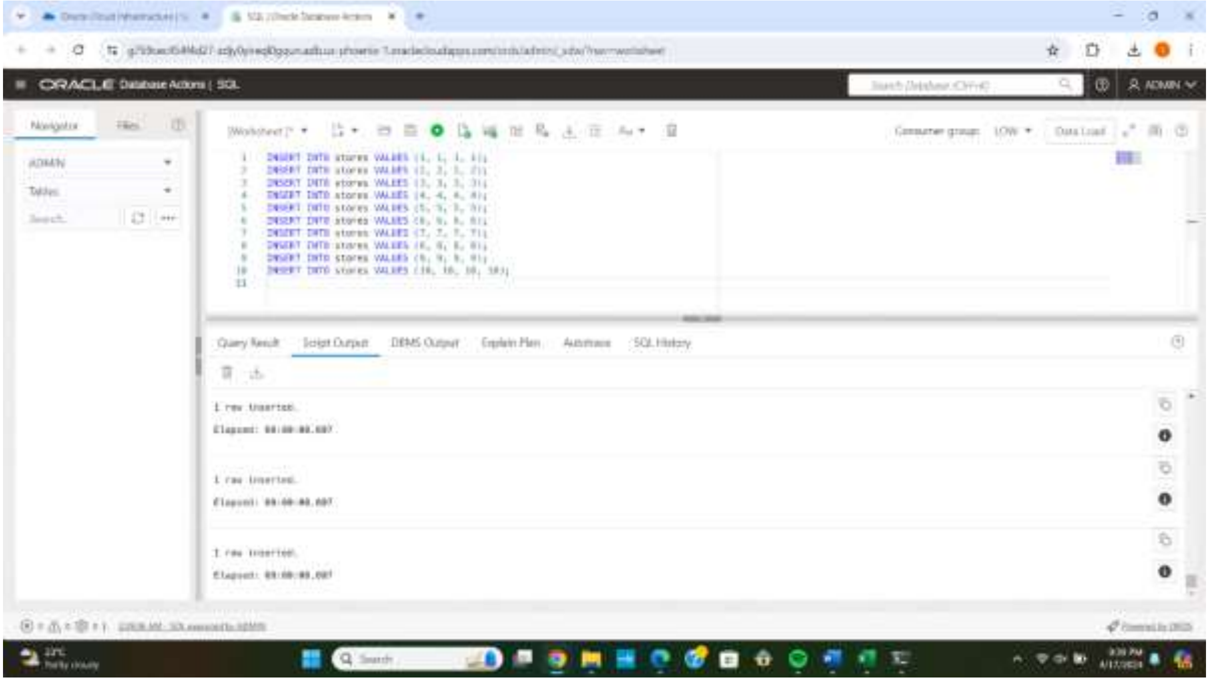
Creating a table named stores



```
CREATE TABLE stores (
    animal_info_id INT REFERENCES animal_info(animal_id),
    training_history_id INT REFERENCES training_history(training_ID),
    grooming_history_id INT REFERENCES grooming_history(grooming_id),
    medical_history_id INT REFERENCES medical_history(medical_id),
    PRIMARY KEY (animal_info_id, training_history_id, grooming_history_id, medical_history_id)
);
```

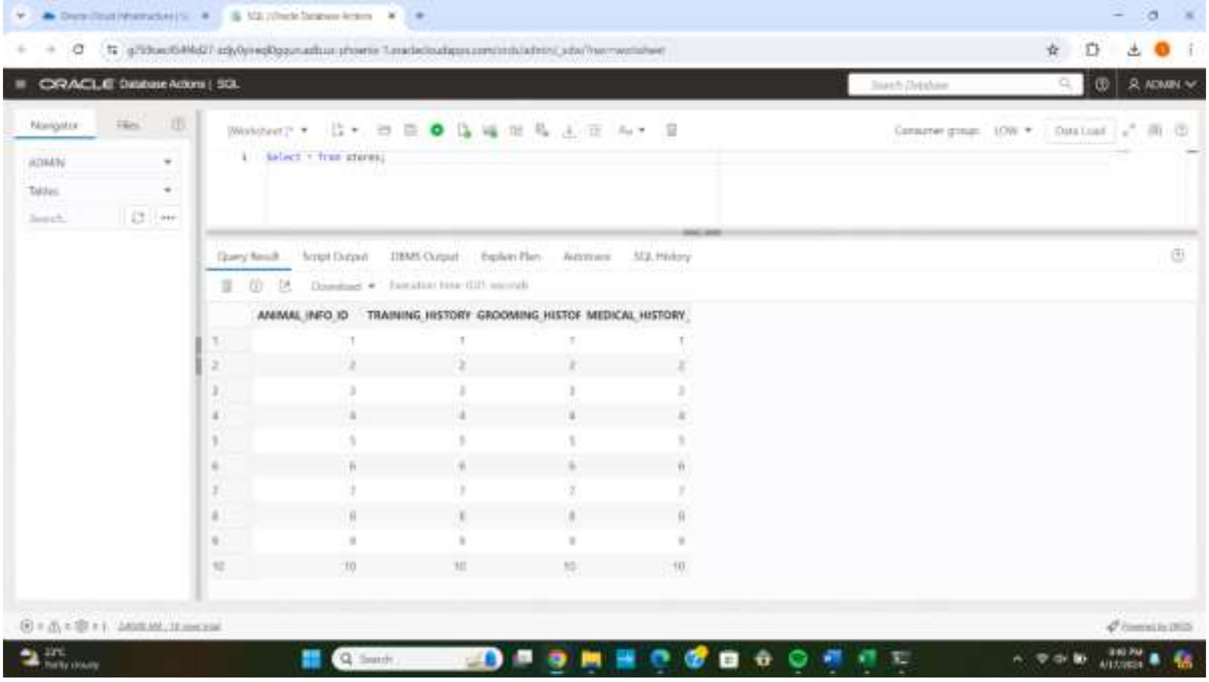
```
1 row inserted.

Elapsed: 00:00:00.007


Table STORES created.

Elapsed: 00:00:00.024
```

Inserting values to stores



```
INSERT INTO stores VALUES (1, 1, 1, 1);
INSERT INTO stores VALUES (2, 2, 2, 2);
INSERT INTO stores VALUES (3, 3, 3, 3);
INSERT INTO stores VALUES (4, 4, 4, 4);
INSERT INTO stores VALUES (5, 5, 5, 5);
INSERT INTO stores VALUES (6, 6, 6, 6);
INSERT INTO stores VALUES (7, 7, 7, 7);
INSERT INTO stores VALUES (8, 8, 8, 8);
INSERT INTO stores VALUES (9, 9, 9, 9);
INSERT INTO stores VALUES (10, 10, 10, 10);
```

```
1 row inserted.

Elapsed: 00:00:00.007


1 row inserted.

Elapsed: 00:00:00.007


1 row inserted.

Elapsed: 00:00:00.007
```



```
select * from stores;
```

	ANIMAL_INFO_ID	TRAINING_HISTORY	GROOMING_HISTOF	MEDICAL_HISTORY
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	5	5
6	6	6	6	6
7	7	7	7	7
8	8	8	8	8
9	9	9	9	9
10	10	10	10	10

INDIVIDUAL CONTRIBUTION:

First,I had an essential role in developing the database for the QUELIFE Animal Shelter, concentrating on the design and building of important tables that would accurately reflect the complex processes of the shelter. My initial work was the' shelter_branch' table, which was thoughtfully designed to capture the physical characteristics and capacity of each shelter, guaranteeing a complete representation of its needs and capabilities.

One thing I was quite passionate about was the'medical_history' table. I created a veterinary record alongside medical experts to create a framework that was both accurate in tracking each animal's health journey and simple enough for employees to update and maintain without specific training. This table is evidence of the shelter's dedication to providing outstanding treatment.

The task of creating the 'adoption_details' table was really exciting. My goal was to build a database that would serve as a smooth connection between adoptive families and animals by combining both practical and emotional components. Every entry was designed to celebrate the beginning of a new chapter in the animals' life and offer a fresh perspective on the adoption procedures.

Apart from this, I also created the foundation for the'shelter_transfer_records' database, which carefully documents animals' transfers between shelters. This made sure the animals had consistent care as they made their way to permanent homes and had a smooth transition.

I added more to the database than just tables; I made sure that every data item and every line of code reflected the values of the shelter. Every table I built was a thoughtful and practical combination, intended to support the shelter's long-term care and conservation goals.