



**Course: Advance Database Management System**

**Sec: B**

**Project Name: Helping Hand Management System**

**Group: 08**

**Group Members**

Name	ID	Contribution	Topic
<b>Tahmid Al Rafid Siddique</b>	<b>20-42086-1</b>	<b>25%</b>	<b>Query Writing, ER Diagram, Data Insertion</b>
<b>Istyak Ahmed</b>	<b>21-45658-3</b>	<b>25%</b>	<b>Diagrams, Interface Design, Scenario Description, Database Connection</b>
<b>Mohammed Adnan Babu</b>	<b>21-45454-3</b>	<b>25%</b>	<b>Normalization, Finalization, Conclusion</b>
<b>Sadakatul Ferdous Pranto</b>	<b>21-45623-3</b>	<b>25%</b>	<b>Schema Diagram, Table Creation, Project Proposal</b>

## Table of Contents

Introduction .....	3
Project Proposal .....	3
Diagrams .....	5
Class Diagram .....	5
Use Case Diagram .....	6
Activity Diagram .....	7
Interface Design .....	8
Scenario Description .....	11
ER Diagram .....	12
Normalization .....	13
Finalization .....	18
SchemaDiagram .....	19
Table Creation .....	20
Data Insertion .....	40
Table: Feedbacks .....	46
Basic Query Writing .....	54
Advanced Query Writing (PL/SQL): .....	78
Conclusion .....	100

## **Project Updates**

- Cardinality issue with scenario description has been attempted to solve.
- Ternary from ER Diagram was removed.
- Slight Changes were made to the Normalization and Table Creation.
- Schema Diagram was updated.

## **Introduction**

Our project revolves around building a Helping Hand Management System (HHMS) to streamline the recruitment process for domestic helpers. The HHMS connects homeowners with verified helpers, simplifying tasks such as profile browsing, job matching, and scheduling. It also supports agencies with profile management and contract tracking. Our aim is to provide an intuitive platform that enhances efficiency and trust, improving the experience for both employers and workers.

## **Project Proposal**

We propose the development of a Home Service Management System (HSMS) to address the operational inefficiencies faced by service agencies in managing interactions between homeowners, helpers, and agency staff. The HSMS aims to streamline workflows, enhance communication, and improve overall service quality. By automating manual tasks and centralizing critical data, the system will create a seamless experience for both service providers and recipients.

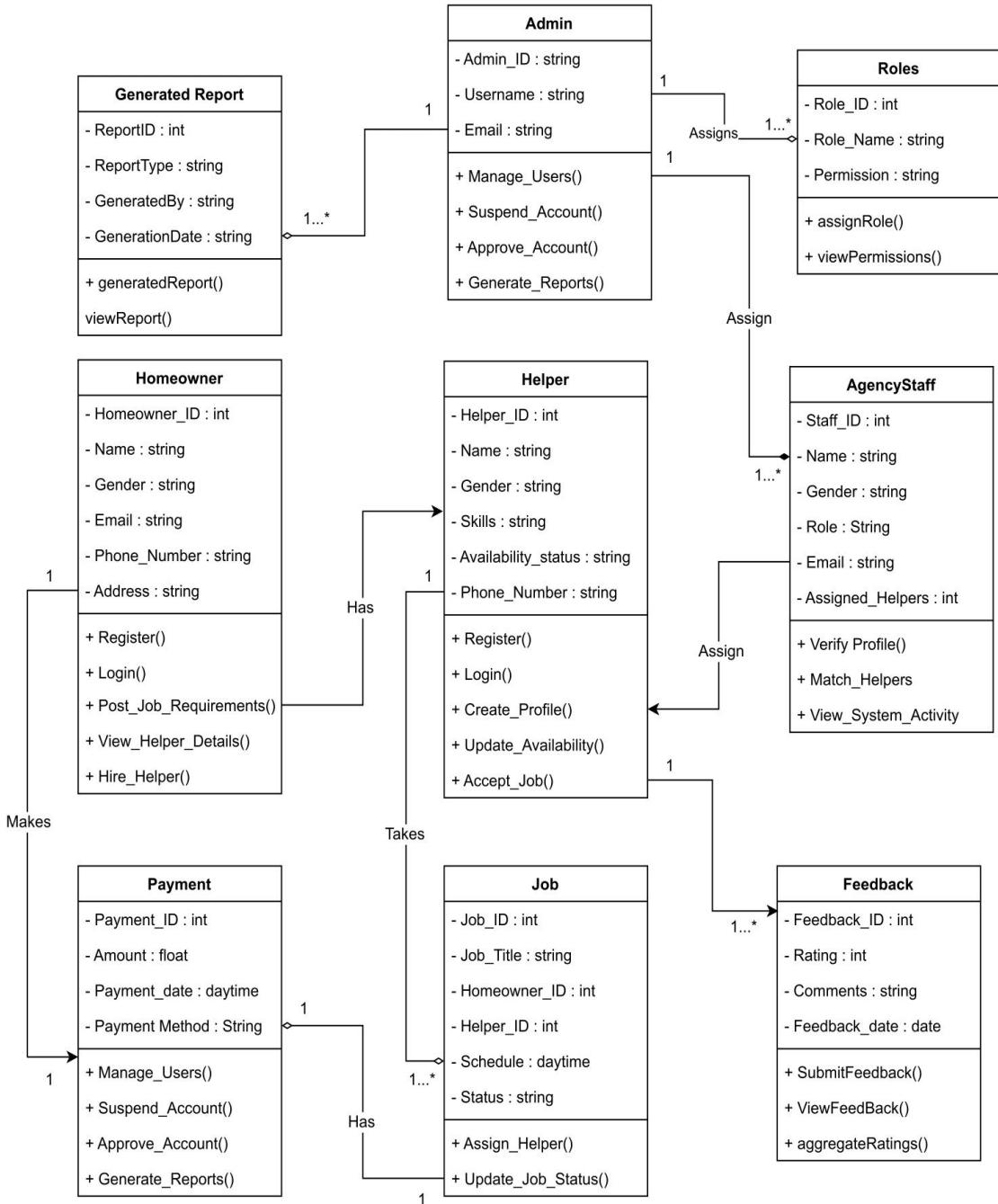
The primary objective of the HSMS is to simplify management tasks and ensure operational transparency while maintaining a user-friendly interface. This system will encompass essential functionalities, such as job management, where homeowners can post and schedule jobs while tracking their status. It will also include tools for user management, ensuring that accurate and secure records are maintained for homeowners, helpers, and administrative staff. Payment management will allow the recording and monitoring of transactions, while feedback and communication modules will facilitate service improvements and timely notifications. Additionally, the system will generate detailed reports to assist agencies in making informed decisions.

To achieve these goals, we will adopt an iterative development approach that emphasizes stakeholder engagement at every stage. We will begin with thorough requirements gathering to align with the specific needs of users. Following this, we will design a robust database structure and implement a scalable, secure system tailored to handle complex workflows efficiently. The implementation phase will focus on developing all modules cohesively, followed by extensive testing to ensure functionality, reliability, and user satisfaction. Finally, we will deploy the system and provide comprehensive training and documentation to ensure smooth adoption by all stakeholders.

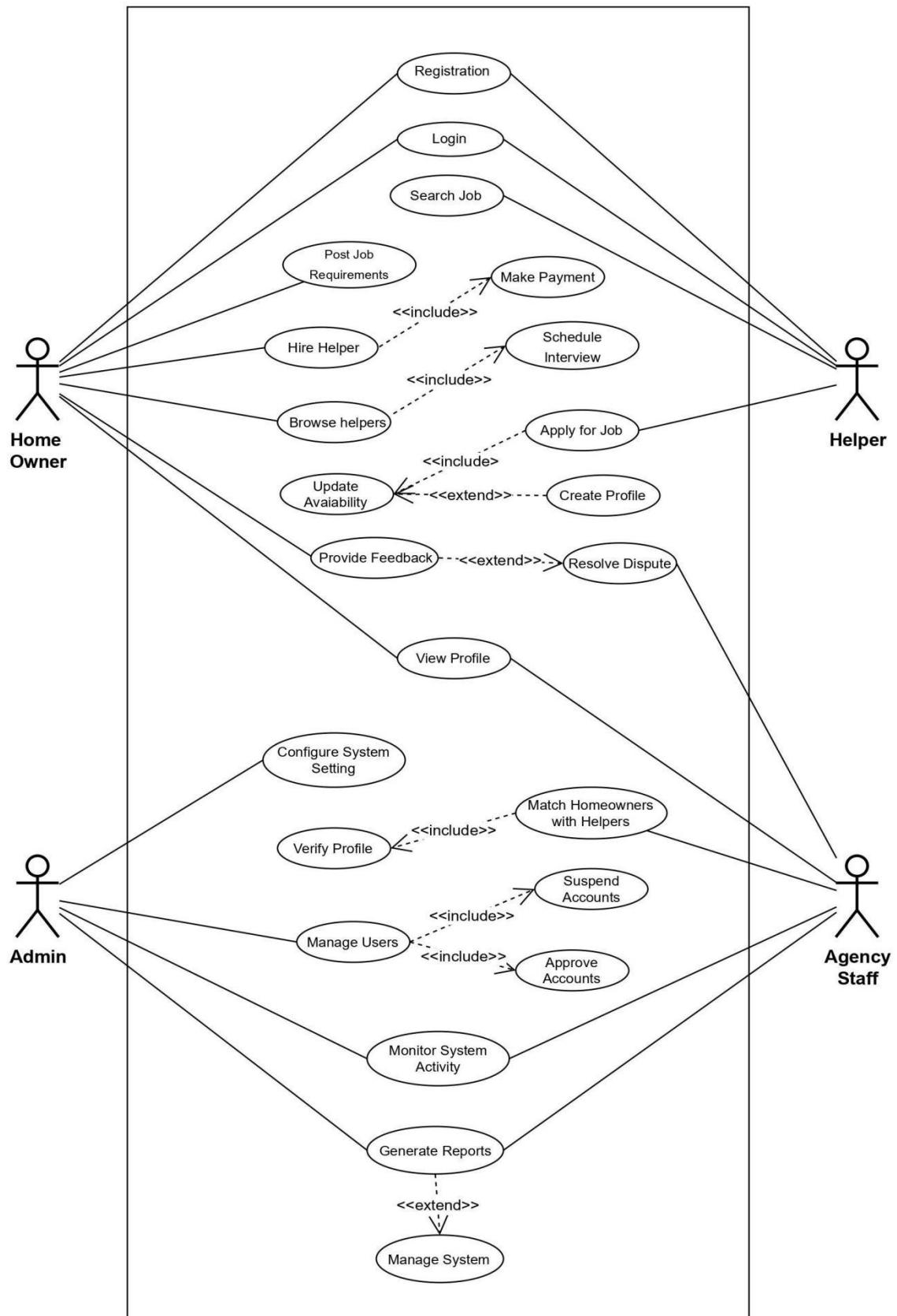
This project will deliver a fully functional Home Service Management System, accompanied by user documentation, a deployment-ready platform, and support for ongoing maintenance. By revolutionizing how service agencies operate, our HSMS will foster automation, improve communication, and significantly enhance the user experience. With a committed team and a clear roadmap, we are confident in delivering a high-quality solution that meets the demands of all involved parties.

# Diagrams

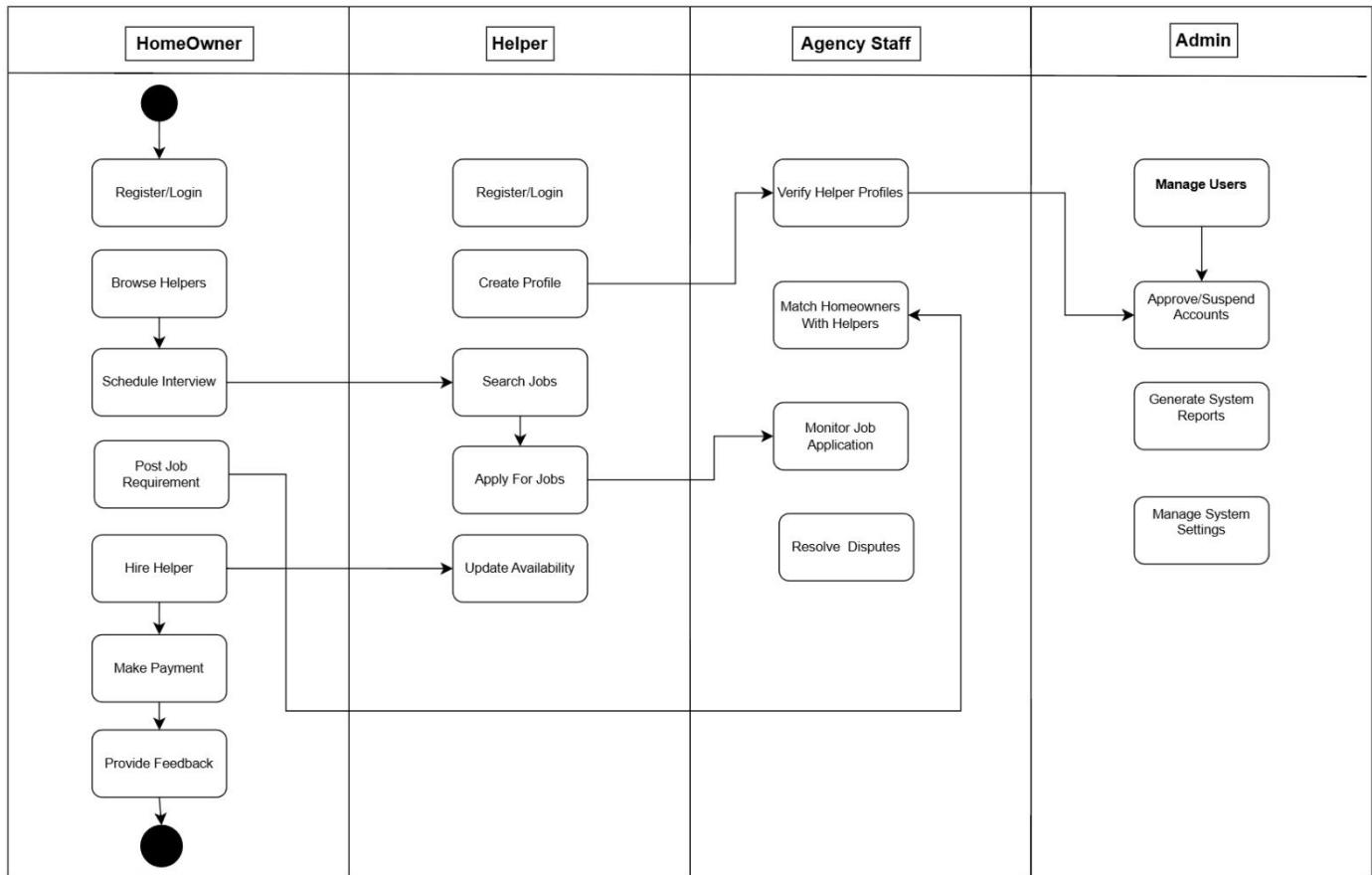
## Class Diagram



## Use Case Diagram

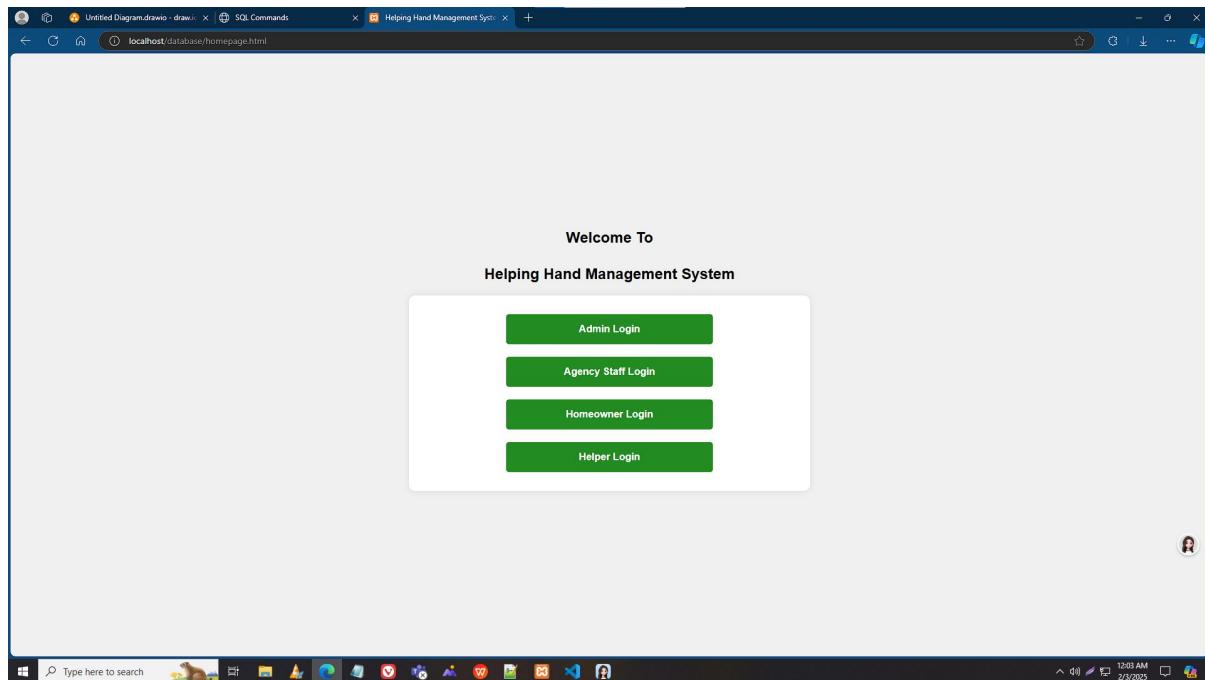


## Activity Diagram

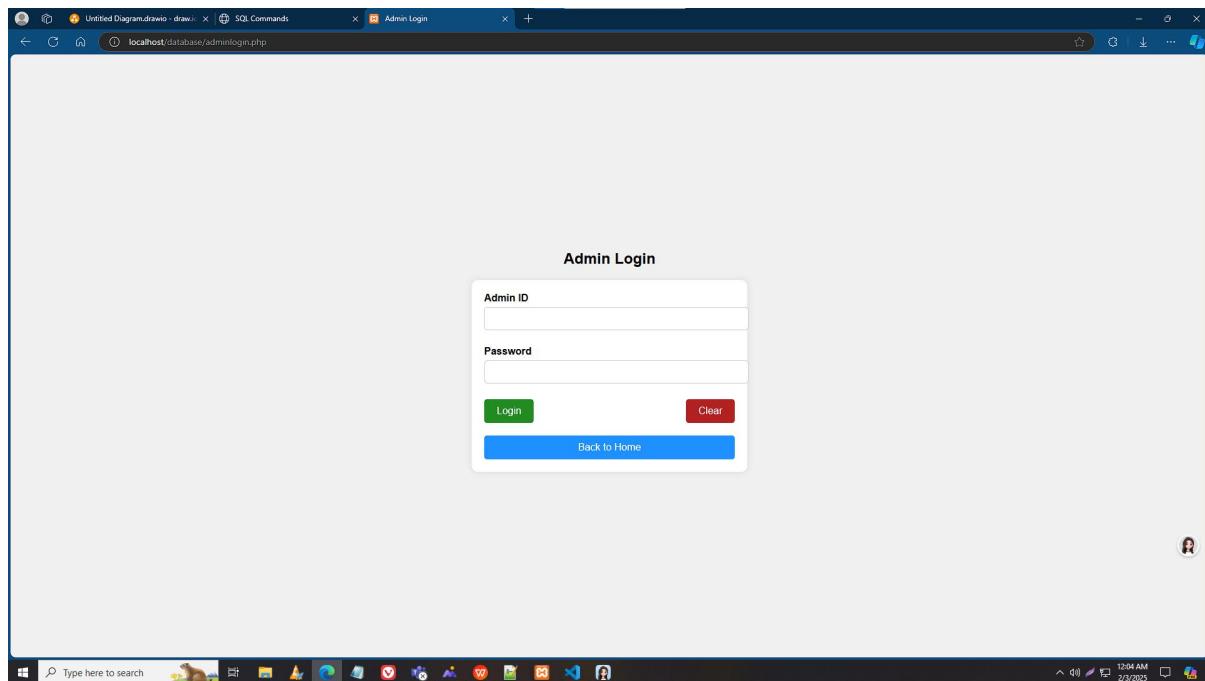


# Interface Design

# Project Homepage



## Admin Login Page



## Admin Dashboard & Data Visualization

Welcome, Admin!

Show & Manage Entities:

- Homeowners [Add Homeowners](#)
- Helpers [Add Helpers](#)
- Agency Staff [Add Agency Staff](#)
- Jobs
- Payments
- Feedbacks
- Reports

P_ID	P_AMOUNT	P_DATE	P_METHOD	PAY_STATUS
1	150	11-APR-24	Credit Card	Completed
2	250	21-APR-24	PayPal	Pending
3	100	16-APR-24	Debit Card	Completed
4	200	13-APR-24	Bank Transfer	Pending
5	300	26-APR-24	Cash	Completed

[Back to Homepage](#)

## Add Homeowners

Add Homeowner

ID:

Name:

Email:

Password:

City:  Street:

House:  Phone:

Gender:  Age:   
Select Gender

[Submit](#) [Reset](#)

[Go Back](#)

## Job Submission Form

The screenshot shows a Windows desktop environment with a browser window open to a job submission form. The title bar of the browser says "Submit Jobs". The form contains fields for "Job ID" (a text input), "Job Title" (a text input), "Status" (a dropdown menu showing "Available"), "Schedule Date" (a date input field with the placeholder "mm/dd/yyyy"), and two buttons: "Submit" (green) and "Reset" (red). Below the form is a blue "Go Back" button. The browser's address bar shows the URL "localhost/database/submitjob.php". The taskbar at the bottom includes icons for various applications like File Explorer, Edge, and File History.

**Submit Jobs**

Job ID:

Job Title:

Status:

Schedule Date:

## Helper Job Application

The screenshot shows a Windows desktop environment with a browser window open to a page listing available jobs for helpers. The title bar of the browser says "Available Jobs for Helpers". The page features two boxes labeled "Job 1" and "Job 2". Each box contains job details and an "Apply" button. Below the boxes is a blue "Go Back" button. The browser's address bar shows the URL "localhost/database/applyjobs.html". The taskbar at the bottom includes icons for various applications like File Explorer, Edge, and File History.

**Available Jobs for Helpers**

**Job 1**

**Job Type:** House Cleaning  
**Homeowner Name:** John Doe  
**Address:** 123 Main St, Cityville  
**Skills Required:** Cleaning, Organizing  
**Payment:** \$15/hour  
**Preferred Time:** 9 AM - 12 PM

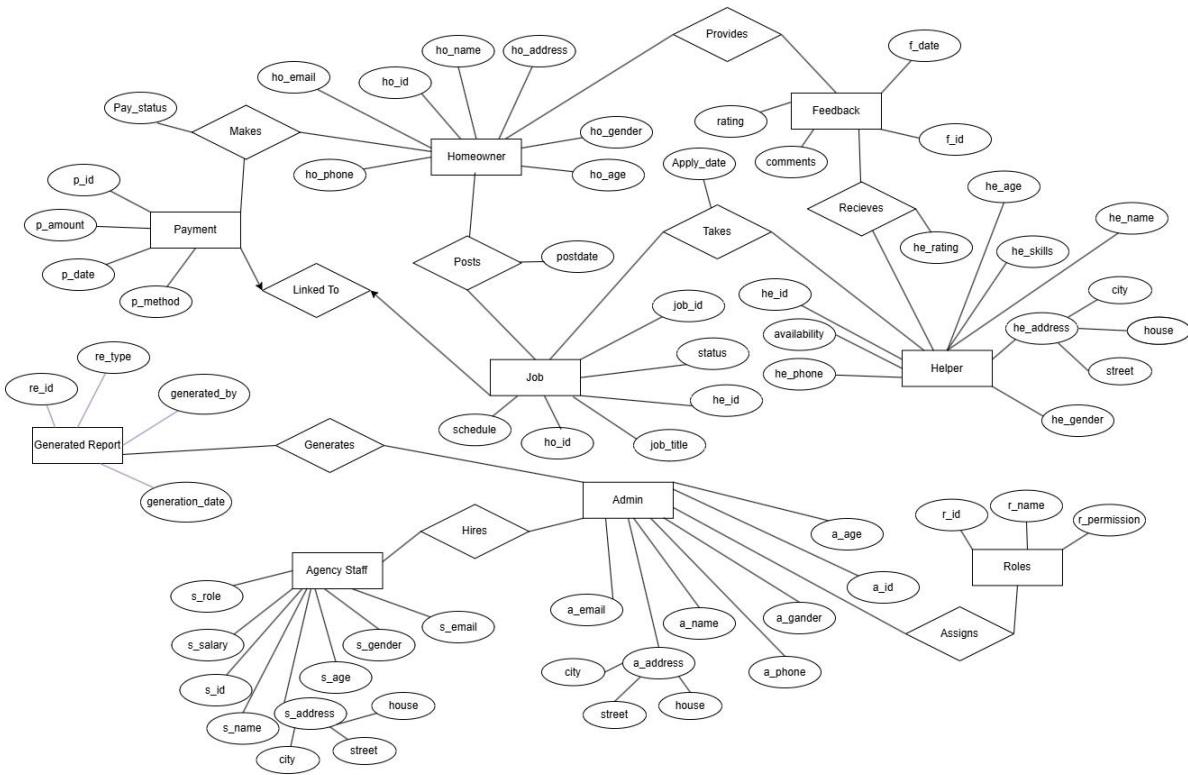
**Job 2**

**Job Type:** Gardening  
**Homeowner Name:** Jane Smith  
**Address:** 456 Oak St, Townsville  
**Skills Required:** Gardening, Landscaping  
**Payment:** \$20/hour  
**Preferred Time:** 2 PM - 5 PM

## **Scenario Description**

The Homeowner-Helper Management System allows homeowners to efficiently hire helpers for various jobs. Each homeowner registers with a unique ID and maintains a profile that includes personal details such as name, age, gender, email, phone number, and address. Homeowners can post multiple job listings, each identified by a unique job ID, containing details like job title, schedule, status, and posting date. Helpers also register with unique IDs and provide information such as name, age, gender, skills, address, phone number, and availability. Helpers can apply for different jobs, and each job can receive applications from multiple helpers. Homeowners have the ability to provide feedback on the helpers they hire. This feedback includes a rating, comments, application date, and feedback date. Since a homeowner can hire different helpers for various jobs, they can provide multiple feedback entries, and each helper can receive feedback from different homeowners. The system tracks and updates the helper's overall rating based on these reviews. Payments for jobs are securely processed, with each payment linked to a specific job and homeowner. Homeowners can make multiple payments, each recorded with details such as payment ID, amount, date, method, and status. The system ensures transparency in financial transactions by keeping a clear record of all payments associated with job postings. To maintain system oversight, agency staff are responsible for generating reports that provide insights into activities within the platform. Reports contain details such as report ID, generation date, and type. Administrators oversee the entire system, managing users, job postings, and financial transactions. Each admin has a profile containing an ID, contact details, demographic information, and assigned roles. Roles define different permissions within the system, and an admin can be assigned multiple roles while each role can be assigned to different admins. By ensuring a well-structured job posting, application, payment, and feedback system, the platform promotes efficient hiring, secure transactions, and effective administrative management, offering a seamless experience for both homeowners and helpers.

# ER Diagram



## Normalization

### Relation: Posts

UNF:

ho\_id, ho\_name, ho\_email, ho\_address, ho\_phone, ho\_gender, ho\_age, job\_id,  
job\_title, status, schedule, postdate

1NF:

ho\_id, ho\_name, ho\_email, ho\_address, ho\_phone, ho\_gender, ho\_age, job\_id,  
job\_title, status, schedule, postdate

2NF

1. ho\_id, ho\_name, ho\_email, ho\_address, ho\_phone, ho\_gender, ho\_age
2. job\_id, ho\_id, job\_title, status, schedule, postdate

3NF

1. ho\_id, ho\_name, ho\_email, ho\_address, ho\_phone, ho\_gender, ho\_age
2. job\_id, ho\_id, job\_title, status, schedule, postdate
3. ho\_id, job\_id

### Relation: Linked To

UNF:

job\_id, job\_title, status, schedule, ho\_id, he\_id, postdate, p\_id, p\_amount, p\_date,  
p\_method, pay\_status

1NF:

job\_id, job\_title, status, schedule, ho\_id, he\_id, postdate, p\_id, p\_amount, p\_date,  
p\_method, pay\_status

2NF:

1. job\_id, job\_title, status, schedule, ho\_id, he\_id, postdate
2. p\_id, job\_id, p\_amount, p\_date, p\_method, pay\_status

3NF:

1. job\_id, job\_title, status, schedule, ho\_id, he\_id, postdate
2. p\_id, job\_id, p\_amount, p\_date, p\_method, pay\_status
3. job\_id, p\_id

### **Relation: Makes**

UNF:

ho\_id, ho\_name, ho\_address (city, street, house), ho\_gender, ho\_phone, p\_id,  
p\_amount, p\_date, p\_method, pay\_status

1NF:

ho\_id, ho\_name, city, street, house, ho\_gender, ho\_phone, p\_id, p\_amount, p\_date,  
p\_method, pay\_status

2NF:

ho\_id, ho\_name, city, street, house, ho\_gender, ho\_phone  
p\_id, p\_amount, p\_date, p\_method, pay\_status, ho\_id

3NF:

ho\_id, ho\_name, city, street, house, ho\_gender, ho\_phone  
p\_id, p\_amount, p\_date, p\_method, pay\_status  
ho\_id, p\_id

### **Relation: Takes**

UNF:

job\_id, status, job\_title, schedule, he\_id, availability, he\_name, he\_address (city, street,  
house), he\_gender, he\_phone

1NF:

job\_id, status, job\_title, schedule, he\_id, availability, he\_name, city, street, house, he\_gender, he\_phone

2NF:

1. he\_id, he\_name, city, street, house, he\_gender, he\_phone, availability
2. job\_id, status, job\_title, schedule, he\_id

3NF:

1. he\_id, he\_name, city, street, house, he\_gender, he\_phone
2. he\_id, availability
3. job\_id, status, job\_title, schedule, he\_id

### **Relation: Provides**

UNF:

ho\_id, ho\_name, ho\_address (city, street, house), ho\_gender, ho\_phone, f\_id, rating, comments, apply\_date

1NF:

ho\_id, ho\_name, city, street, house, ho\_gender, ho\_phone, f\_id, rating, comments, apply\_date

2NF:

1. ho\_id, ho\_name, city, street, house, ho\_gender, ho\_phone
2. f\_id, rating, comments, apply\_date, ho\_id

3NF:

1. ho\_id, ho\_name, city, street, house, ho\_gender, ho\_phone
2. f\_id, rating, comments, apply\_date
3. ho\_id, f\_id

### **Relation: Receives**

UNF:

he\_id, he\_name, he\_address (city, street, house), he\_gender, he\_phone, f\_id, f\_date, rating, comments

1NF:

he\_id, he\_name, city, street, house, he\_gender, he\_phone, f\_id, f\_date, rating, comments

2NF:

1. he\_id, he\_name, city, street, house, he\_gender, he\_phone
2. f\_id, f\_date, rating, comments, he\_id

3NF:

1. he\_id, he\_name, city, street, house, he\_gender, he\_phone
2. f\_id, f\_date, rating, comments
3. he\_id, f\_id

### **Relation: Assigns**

UNF:

s\_id, s\_name, s\_address (city, street, house), s\_gender, s\_email, r\_id, r\_name, r\_permission

1NF:

s\_id, s\_name, city, street, house, s\_gender, s\_email, r\_id, r\_name, r\_permission

2NF:

1. s\_id, s\_name, city, street, house, s\_gender, s\_email
2. r\_id, r\_name, r\_permission, s\_id

3NF:

1. s\_id, s\_name, city, street, house, s\_gender, s\_email
2. r\_id, r\_name, r\_permission
3. s\_id, r\_id

### **Relation: Generates**

UNF:

s\_id, s\_name, s\_address (city, street, house), s\_gender, s\_email, re\_id, re\_type, generated\_by, generation\_date

1NF:

s\_id, s\_name, city, street, house, s\_gender, s\_email, re\_id, re\_type, generated\_by, generation\_date

2NF:

1. s\_id, s\_name, city, street, house, s\_gender, s\_email
2. re\_id, re\_type, generated\_by, generation\_date, s\_id

3NF:

1. s\_id, s\_name, city, street, house, s\_gender, s\_email
2. re\_id, re\_type, generated\_by, generation\_date
3. s\_id, re\_id

### **Relation: Hires**

UNF:

a\_id, a\_name, a\_email, a\_phone, a\_address (city, street, house), a\_age, a\_gender, s\_id, s\_name, s\_email, s\_role, s\_salary, s\_age, s\_gender, s\_address (city, street, house)

1NF:

a\_id, a\_name, a\_email, a\_phone, a\_address (city, street, house), a\_age, a\_gender, s\_id, s\_name, s\_email, s\_role, s\_salary, s\_age, s\_gender, s\_address (city, street, house)

2NF:

a\_id, a\_name, a\_email, a\_phone, a\_street, a\_house, a\_city, a\_age, a\_gender  
s\_id, s\_name, s\_email, s\_role, s\_salary, s\_age, s\_gender, s\_street, s\_house, s\_city

3NF:

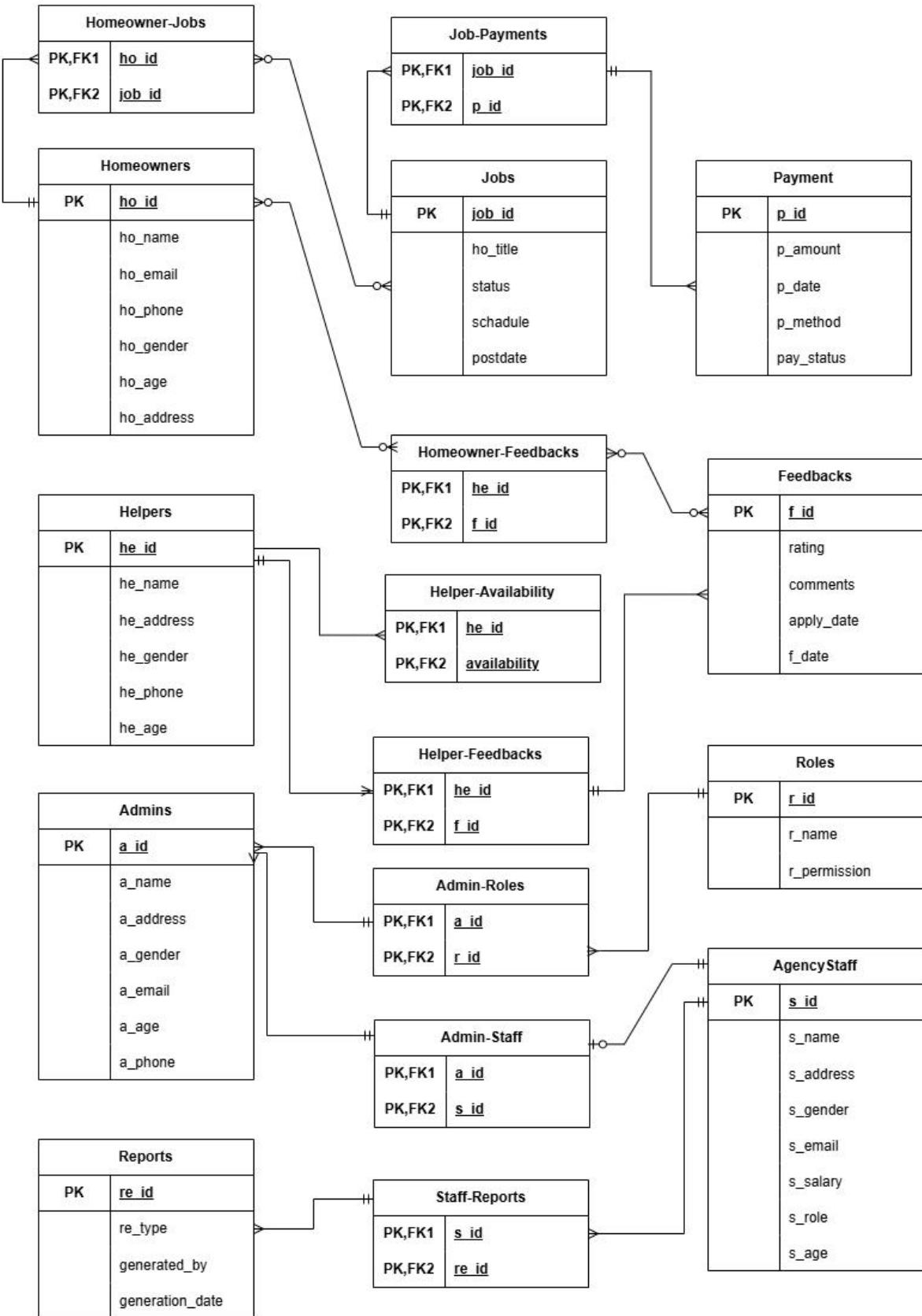
a\_id, a\_name, a\_email, a\_phone, a\_age, a\_gender, address\_id  
s\_id, s\_name, s\_email, s\_role, s\_salary, s\_age, s\_gender, address\_id  
a\_id, s\_id

# **Finalization**

## **Tables:**

1. ho\_id, ho\_name, ho\_email, ho\_address (city, street, house), ho\_phone, ho\_gender, ho\_age
2. job\_id, ho\_id, job\_title, status, schedule, postdate
3. ho\_id, job\_id
4. p\_id, p\_amount, p\_date, p\_method, pay\_status
5. job\_id, p\_id
6. he\_id, he\_name, he\_address (city, street, house), he\_gender, he\_phone
7. he\_id, availability
8. f\_id, rating, comments, apply\_date, f\_date
9. ho\_id, f\_id
10. he\_id, f\_id
11. r\_id, r\_name, r\_permission
12. a\_id, a\_name, a\_address (city, street, house), a\_gender, a\_phone, a\_email, a\_age
13. a\_id, r\_id
14. s\_id, s\_name, s\_address (city, street, house), s\_gender, s\_email, s\_salary, s\_role, s\_age
15. re\_id, re\_type, generated\_by, generation\_date
16. s\_id, re\_id
17. a\_id, s\_id

# SchemaDiagram



## Table Creation

### Table: Homeowners

Sequence: CREATE SEQUENCE homeowners\_seq START WITH 10001  
INCREMENT BY 1 NOCACHE NOCYCLE;

Table Query:

```
CREATE TABLE Homeowners (
    ho_id NUMBER(10) PRIMARY KEY,
    ho_name VARCHAR2(100) NOT NULL,
    ho_email VARCHAR2(100) UNIQUE,
    ho_password VARCHAR2(255) NOT NULL,
    ho_address_city VARCHAR2(100),
    ho_address_street VARCHAR2(100),
    ho_address_house VARCHAR2(100),
    ho_phone VARCHAR2(15) UNIQUE,
    ho_gender CHAR(1),
    ho_age NUMBER(3)
);
```

The screenshot shows the Oracle Database Express Edition SQL Commands window. In the SQL editor, the following SQL code is entered:

```
CREATE TABLE Homeowners (
    ho_id NUMBER(10) PRIMARY KEY,
    ho_name VARCHAR2(100) NOT NULL,
    ho_email VARCHAR2(100) UNIQUE,
    ho_address CITY VARCHAR2(100),
    ho_address_street VARCHAR2(100),
    ho_address_house VARCHAR2(100),
    ho_phone VARCHAR2(15) UNIQUE,
    ho_zipcode CHAR(5),
    ho_age NUMBER(3)
);
```

Below the code, the results of the execution are displayed:

Table created.  
0.04 seconds

Language: en-us Application Express 2.1.0.0.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

Index: CREATE INDEX homeowners\_idx\_email ON Homeowners(ho\_id, ho\_email);

**Results Explain Describe Saved SQL History**

Index created.

0.00 seconds

### Table: Jobs

Sequence: CREATE SEQUENCE jobs\_seq START WITH 20001 INCREMENT BY 1 NOCACHE NOCYCLE;

**Results Explain Describe Saved SQL History**

Sequence created.

0.00 seconds

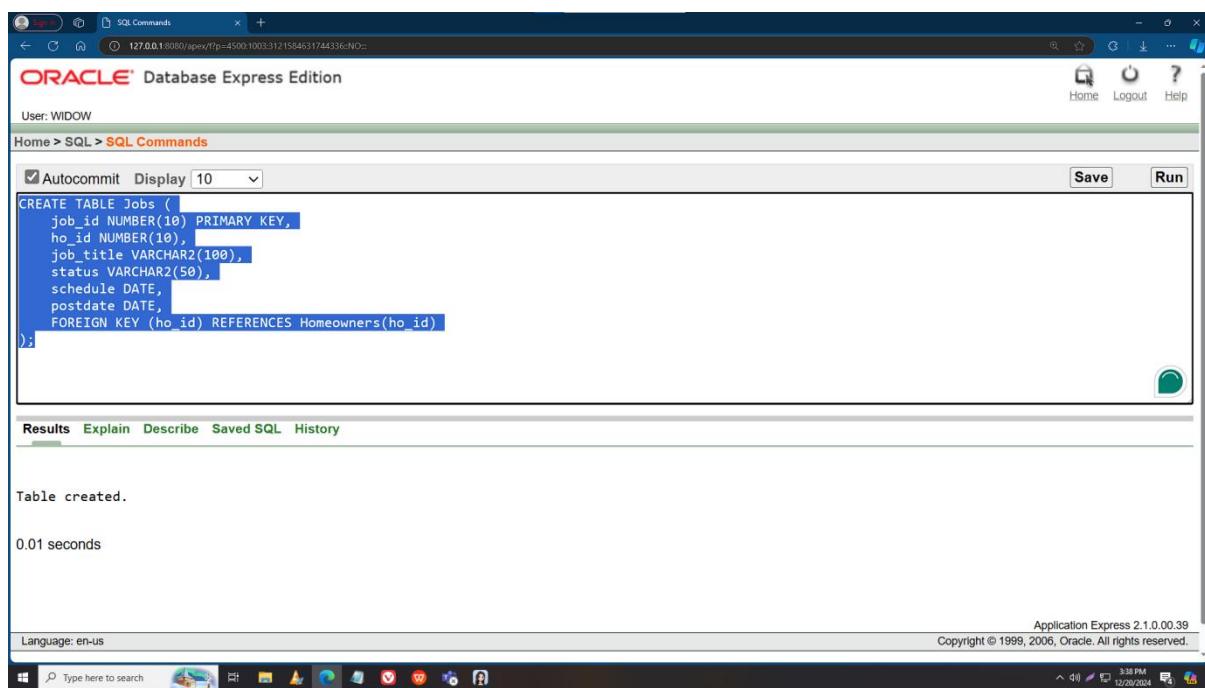
Table Query:

```

CREATE TABLE Jobs (
    job_id NUMBER(10) PRIMARY KEY,
    ho_id NUMBER(10),
    job_title VARCHAR2(100),
    status VARCHAR2(50),
    schedule DATE,
    postdate DATE,
    FOREIGN KEY (ho_id) REFERENCES Homeowners(ho_id));

```

Index: CREATE INDEX jobs\_idx\_title ON Jobs(job\_id, job\_title);



The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL editor window contains the following code:

```

CREATE TABLE Jobs (
    job_id NUMBER(10) PRIMARY KEY,
    ho_id NUMBER(10),
    job_title VARCHAR2(100),
    status VARCHAR2(50),
    schedule DATE,
    postdate DATE,
    FOREIGN KEY (ho_id) REFERENCES Homeowners(ho_id)
);

```

The 'Run' button is highlighted. Below the editor, the results pane shows:

```

Table created.

0.01 seconds

```

The bottom status bar indicates the application version is Application Express 2.1.0.00.39 and the copyright year is 1999, 2006, Oracle. All rights reserved.

### Table: Homeowner-Jobs

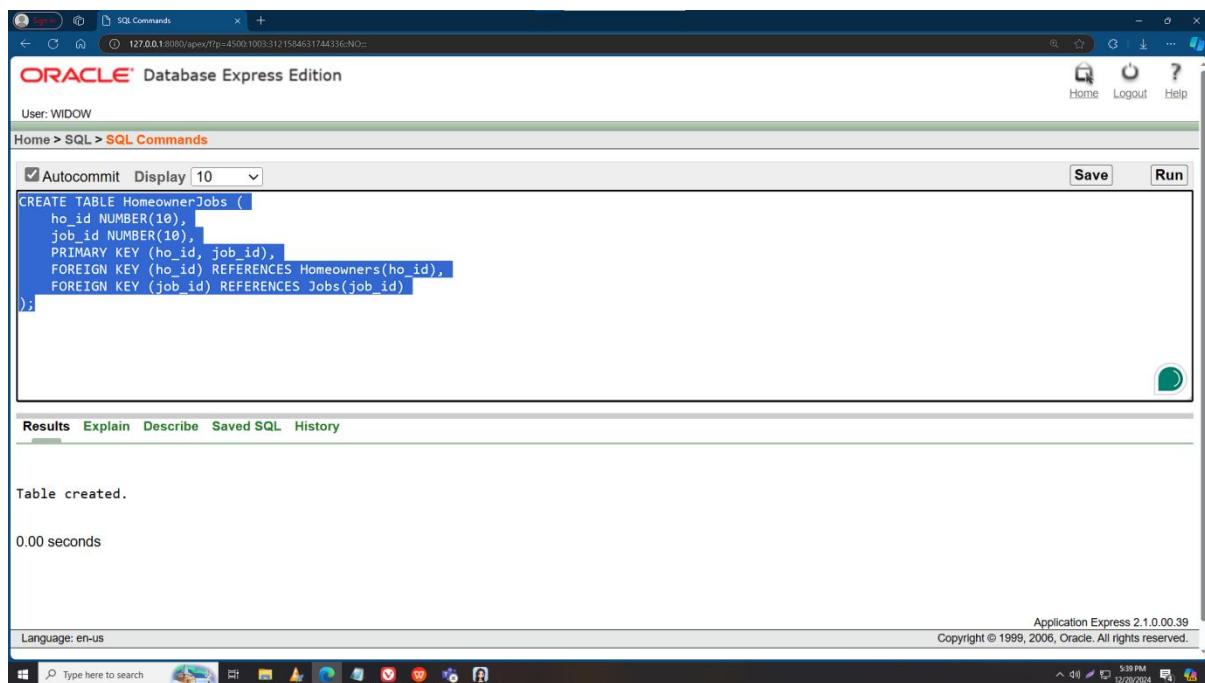
Table Query:

```

CREATE TABLE HomeownerJobs (
    ho_id NUMBER(10),
    job_id NUMBER(10),
    PRIMARY KEY (ho_id, job_id),
    FOREIGN KEY (ho_id) REFERENCES Homeowners(ho_id),
    FOREIGN KEY (job_id) REFERENCES Jobs(job_id)
);

```

Index: CREATE INDEX homeownerjobs\_idx ON HomeownerJobs(ho\_id, job\_id);



The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command entered is:

```

CREATE TABLE HomeownerJobs (
    ho_id NUMBER(10),
    job_id NUMBER(10),
    PRIMARY KEY (ho_id, job_id),
    FOREIGN KEY (ho_id) REFERENCES Homeowners(ho_id),
    FOREIGN KEY (job_id) REFERENCES Jobs(job_id)
);

```

The results pane shows the output:

```

Table created.

0.00 seconds

```

At the bottom, it says "Language: en-us" and "Application Express 2.1.0.0.39 Copyright © 1989, 2006, Oracle. All rights reserved."

### Table: Payments

Sequence: CREATE SEQUENCE payments\_seq START WITH 30001 INCREMENT BY 1 NOCACHE NOCYCLE;



The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command entered is:

```

CREATE SEQUENCE payments_seq
    START WITH 30001
    INCREMENT BY 1
    NOCACHE
    NOCYCLE;

```

The results pane shows the output:

```

Sequence created.

0.00 seconds

```

Table Query:

```

CREATE TABLE Payments (
    p_id NUMBER(10) PRIMARY KEY,
    p_amount DECIMAL(10, 2),
    p_date DATE,
    p_method VARCHAR2(50),
    pay_status VARCHAR2(50)
);

```

The screenshot shows the Oracle Database Express Edition interface. In the top navigation bar, it says "User WIDOW" and "ORACLE Database Express Edition". Below the navigation bar, there's a toolbar with icons for Home, Logout, and Help. The main area is titled "SQL Commands" and contains a code editor with the following SQL statement:

```

CREATE TABLE Payments (
    p_id NUMBER(10) PRIMARY KEY,
    p_amount DECIMAL(10, 2),
    p_date DATE,
    p_method VARCHAR2(50),
    pay_status VARCHAR2(50)
);

```

Below the code editor, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected. The output section displays the message "Table created." and "0.02 seconds". At the bottom of the window, it says "Language: en-us" and "Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved." The system tray at the bottom of the screen shows the date and time as "12/20/2004 1:41 PM".

Index: CREATE INDEX payments\_idx\_status ON Payments(p\_id, pay\_status);

The screenshot shows the Oracle Database Express Edition interface. The top navigation bar and toolbar are identical to the previous screenshot. The main area is titled "SQL Commands" and contains the following SQL statement:

```

CREATE INDEX payments_idx_status ON Payments(p_id, pay_status);

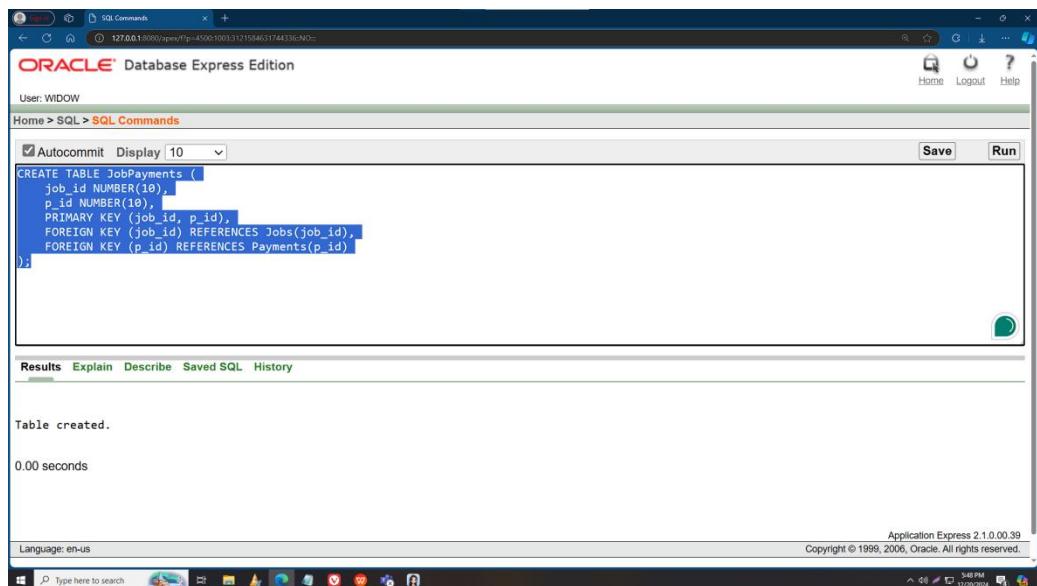
```

The Results tab is selected, and the output section displays the message "Index created." and "0.00 seconds". The bottom of the window shows the same copyright information and system tray as the previous screenshot.

## Table: Job-Payments

Table Query:

```
CREATE TABLE JobPayments (
    job_id NUMBER(10),
    p_id NUMBER(10),
    PRIMARY KEY (job_id, p_id),
    FOREIGN KEY (job_id) REFERENCES Jobs(job_id),
    FOREIGN KEY (p_id) REFERENCES Payments(p_id));
```



The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command to create the JobPayments table is entered in the main pane. The table is defined with two columns: job\_id and p\_id, both of type NUMBER(10). It features a composite primary key (job\_id, p\_id), and foreign keys referencing the Jobs and Payments tables respectively. After executing the command, a message 'Table created.' is displayed in the results pane. The application version is Application Express 2.1.0.0.39.

Index: CREATE INDEX jobpayments\_idx ON JobPayments(job\_id, p\_id);



The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command to create the jobpayments\_idx index is entered in the main pane. The index is created on the JobPayments table, specifically on the columns job\_id and p\_id. After executing the command, a message 'Index created.' is displayed in the results pane. The application version is Application Express 2.1.0.0.39.

## Table: Helpers

Sequence: CREATE SEQUENCE helpers\_seq START WITH 40001 INCREMENT BY 1 NOCACHE NOCYCLE;

Results	Explain	Describe	Saved SQL	History
Sequence created.				

0.00 seconds

Table Query:

```
CREATE TABLE Helpers (
    he_id NUMBER(10) PRIMARY KEY,
    he_name VARCHAR2(100),
    he_password VARCHAR2(255) NOT NULL,
    he_address_city VARCHAR2(100),
    he_address_street VARCHAR2(100),
    he_address_house VARCHAR2(100),
    he_gender CHAR(1),
    he_phone VARCHAR2(15) UNIQUE
);
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. In the SQL pane, the following SQL code is entered:

```
CREATE TABLE Helpers (
    he_id NUMBER(10) PRIMARY KEY,
    he_name VARCHAR2(100),
    he_address_city VARCHAR2(100),
    he_address_street VARCHAR2(100),
    he_address_house VARCHAR2(100),
    he_gender CHAR(1),
    he_phone VARCHAR2(15) UNIQUE
);
```

The Results pane displays the output:

```
Table created.  
0.01 seconds
```

The status bar at the bottom right indicates Application Express 2.1.0.00.39 and Copyright © 1999, 2006, Oracle. All rights reserved.

Index: CREATE INDEX helpers\_idx\_phone ON Helpers(he\_id, he\_phone);

The screenshot shows the Oracle Database Express Edition SQL Commands interface. In the SQL pane, the following SQL code is entered:

```
CREATE INDEX helpers_idx_phone ON Helpers(he_id, he_phone);
```

The Results pane displays the output:

```
Index created.  
0.00 seconds
```

### Table: helper-Availability

Table Query:

```
CREATE TABLE Helper_Availability (
    he_id NUMBER(10),
    availability VARCHAR2(50),
    PRIMARY KEY (he_id, availability),
    FOREIGN KEY (he_id) REFERENCES Helpers(he_id)
);
```

The screenshot shows the Oracle Database Express Edition SQL Commands window. The SQL code entered is:

```
CREATE TABLE HelperAvailability (
    he_id NUMBER(10),
    availability VARCHAR2(50),
    PRIMARY KEY (he_id, availability),
    FOREIGN KEY (he_id) REFERENCES Helpers(he_id)
);
```

The results pane shows the output:

```
Table created.  
0.00 seconds
```

At the bottom, it says "Language: en-us" and "Application Express 2.1 0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved."

Index: CREATE INDEX helperavailability\_idx ON HelperAvailability(he\_id, availability);

The screenshot shows the Oracle Database Express Edition SQL Commands window. The results pane shows the output:

```
Index created.  
0.00 seconds
```

### Table: Feedbacks

Sequence:

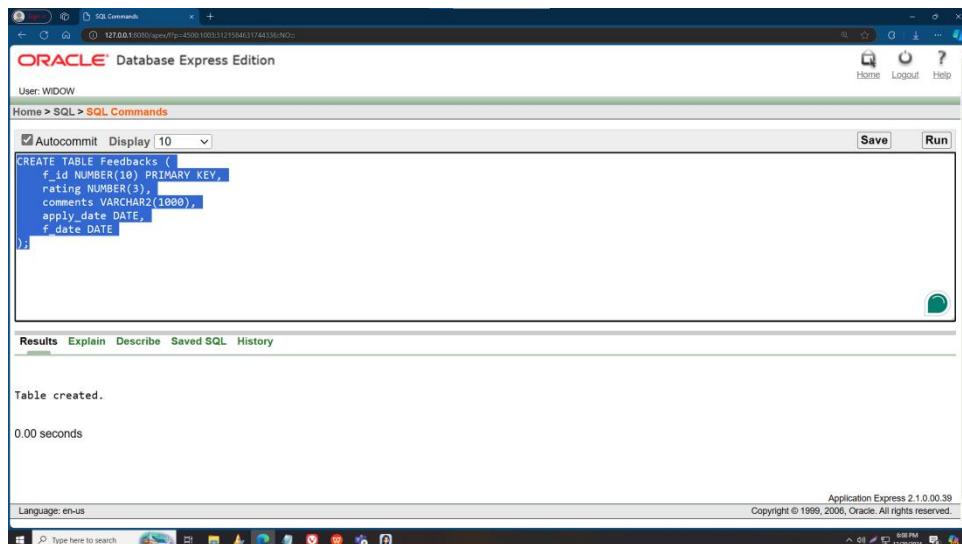
```
CREATE SEQUENCE feedbacks_seq START WITH 50001 INCREMENT BY 1  
NOCACHE NOCYCLE;
```

The screenshot shows the Oracle Database Express Edition SQL Commands window. The results pane shows the output:

```
Sequence created.  
0.00 seconds
```

Table Query:

```
CREATE TABLE Feedbacks (
    f_id NUMBER(10) PRIMARY KEY,
    rating NUMBER(3),
    comments VARCHAR2(1000),
    apply_date DATE,
    f_date DATE);
```



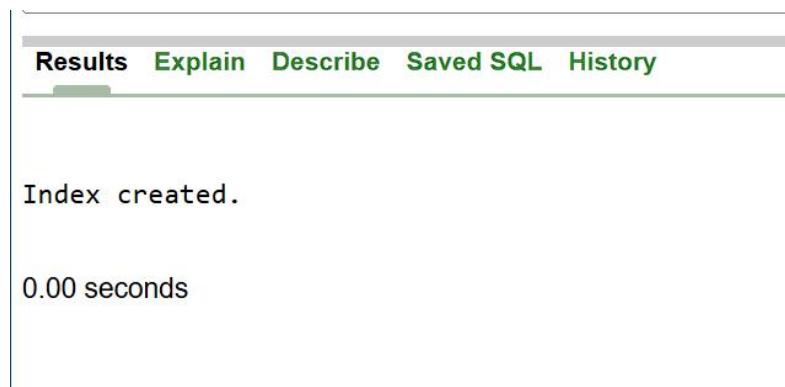
The screenshot shows the Oracle Database Express Edition interface. In the top navigation bar, it says "User WIDOW" and "Home > SQL > SQL Commands". The main area contains the SQL code for creating the Feedbacks table. Below the code, the message "Table created." is displayed, followed by "0.00 seconds". At the bottom right, it says "Application Express 2.1.0.0.39 Copyright © 1995, 2008, Oracle. All rights reserved."

```
CREATE TABLE Feedbacks (
    f_id NUMBER(10) PRIMARY KEY,
    rating NUMBER(3),
    comments VARCHAR2(1000),
    apply_date DATE,
    f_date DATE);

```

Table created.  
0.00 seconds

Index: CREATE INDEX feedbacks\_idx\_rating ON Feedbacks(f\_id, rating);



The screenshot shows the Oracle Database Express Edition interface. In the top navigation bar, it says "User WIDOW" and "Home > SQL > SQL Commands". The main area contains the SQL code for creating an index on the Feedbacks table. Below the code, the message "Index created." is displayed, followed by "0.00 seconds". At the bottom right, it says "Application Express 2.1.0.0.39 Copyright © 1995, 2008, Oracle. All rights reserved."

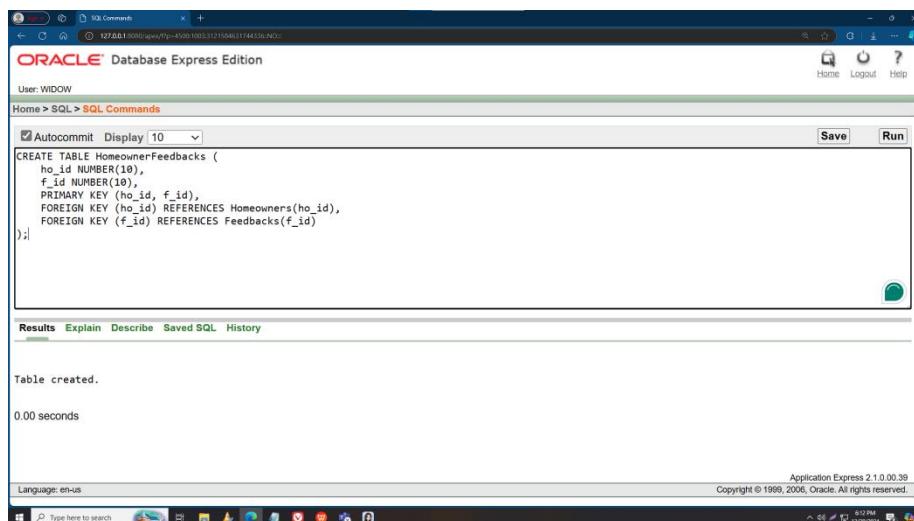
```
CREATE INDEX feedbacks_idx_rating ON Feedbacks(f_id, rating);
```

Index created.  
0.00 seconds

### Table: Homeowner-Feedbacks

Table Query:

```
CREATE TABLE HomeownerFeedbacks (
    ho_id NUMBER(10),
    f_id NUMBER(10),
    PRIMARY KEY (ho_id, f_id),
    FOREIGN KEY (ho_id) REFERENCES Homeowners(ho_id),
    FOREIGN KEY (f_id) REFERENCES Feedbacks(f_id)
);
```



The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a CREATE TABLE statement is entered:

```
CREATE TABLE HomeownerFeedbacks (
    ho_id NUMBER(10),
    f_id NUMBER(10),
    PRIMARY KEY (ho_id, f_id),
    FOREIGN KEY (ho_id) REFERENCES Homeowners(ho_id),
    FOREIGN KEY (f_id) REFERENCES Feedbacks(f_id)
);
```

After running the query, the results show:

```
Table created.  
0.00 seconds
```

The bottom status bar indicates Application Express 2.1.0.0.39 and Copyright © 1999, 2006, Oracle. All rights reserved.

Index: CREATE INDEX homeownerfeedbacks\_idx ON HomeownerFeedbacks(ho\_id, f\_id);



The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a CREATE INDEX statement is entered:

```
CREATE INDEX homeownerfeedbacks_idx ON HomeownerFeedbacks(ho_id, f_id);
```

After running the query, the results show:

```
Index created.  
0.00 seconds
```

### **Table: Helper-Feedback**

Table Query:

```
CREATE TABLE HelperFeedbacks (
    he_id NUMBER(10),
    f_id NUMBER(10),
    PRIMARY KEY (he_id, f_id),
    FOREIGN KEY (he_id) REFERENCES Helpers(he_id),
    FOREIGN KEY (f_id) REFERENCES Feedbacks(f_id)
);
```

Index: CREATE INDEX helperfeedbacks\_idx ON HelperFeedbacks(he\_id, f\_id);

Results	Explain	Describe	Saved SQL	History
Index created.				

### **Table: Roles**

Sequence:

```
CREATE SEQUENCE roles_seq START WITH 60001 INCREMENT BY 1
NOCACHE NOCYCLE;
```

Results	Explain	Describe	Saved SQL	History
Sequence created.				

Table Query:

```

CREATE TABLE Roles (
    r_id NUMBER(10) PRIMARY KEY,
    r_name VARCHAR2(50),
    r_permission VARCHAR2(50)
);

```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the following SQL code is entered:

```

CREATE TABLE Roles (
    r_id NUMBER(10) PRIMARY KEY,
    r_name VARCHAR2(50),
    r_permission VARCHAR2(50)
);

```

After running the command, the results show:

Table created.  
0.00 seconds

At the bottom, it says Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved.

**Index:** CREATE INDEX roles\_idx\_name ON Roles(r\_id, r\_name);

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the following SQL code is entered:

```

CREATE INDEX roles_idx_name ON Roles(r_id, r_name);

```

After running the command, the results show:

Index created.  
0.00 seconds

**Table: Admin**

Sequence:

```

CREATE SEQUENCE admins_seq START WITH 70001 INCREMENT BY 1
NOCACHE NOCYCLE;

```

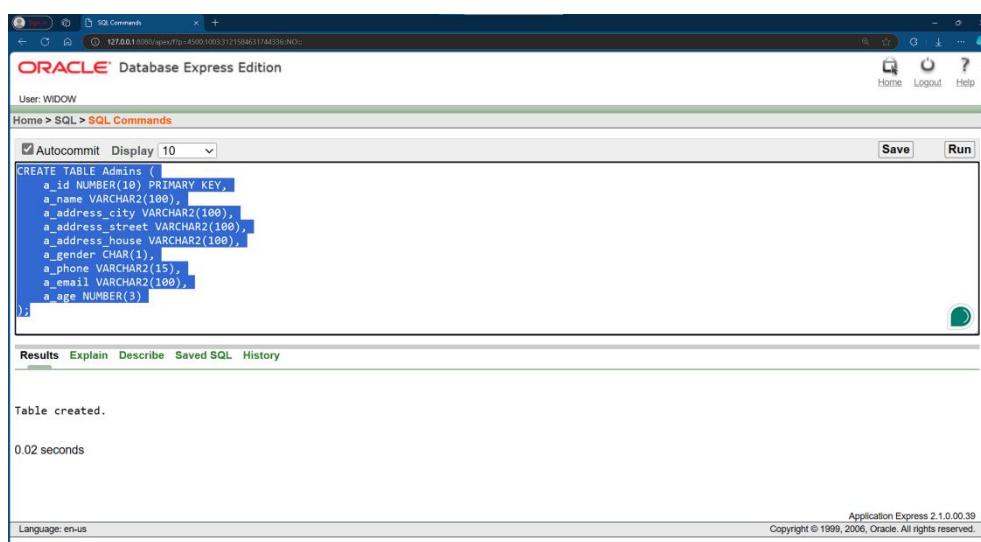
Results Explain Describe Saved SQL History

Sequence created.

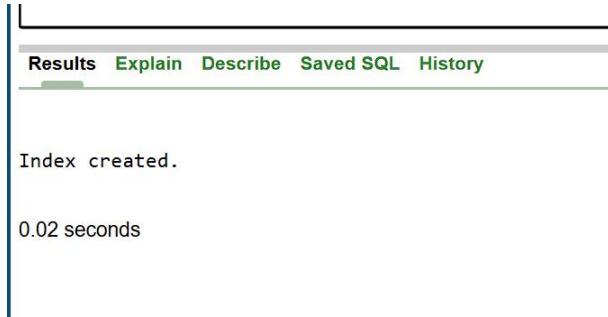
0.00 seconds

## Table Query:

```
CREATE TABLE Admins (
    a_id NUMBER(10) PRIMARY KEY,
    a_name VARCHAR2(100),
    a_password VARCHAR2(255) NOT NULL,
    a_address_city VARCHAR2(100),
    a_address_street VARCHAR2(100),
    a_address_house VARCHAR2(100),
    a_gender CHAR(1),
    a_phone VARCHAR2(15) UNIQUE,
    a_email VARCHAR2(100) UNIQUE,
    a_age NUMBER(3)
);
```



Index: CREATE INDEX admins\_idx\_email ON Admins(a\_id, a\_email);

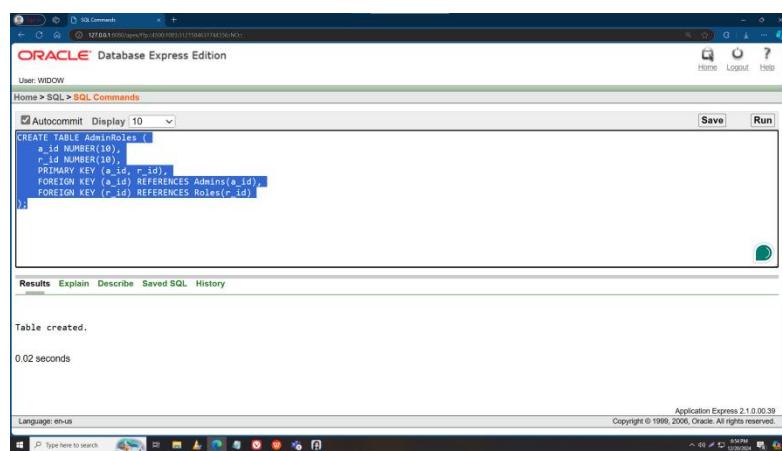


A screenshot of the Oracle SQL Developer interface. The top menu bar shows 'Results' as the active tab. Below the menu, the message 'Index created.' is displayed, followed by '0.02 seconds'.

### Table: Admin-Roles

#### Table Query:

```
CREATE TABLE AdminRoles (
    a_id NUMBER(10),
    r_id NUMBER(10),
    PRIMARY KEY (a_id, r_id),
    FOREIGN KEY (a_id) REFERENCES Admins(a_id),
    FOREIGN KEY (r_id) REFERENCES Roles(r_id));
```



A screenshot of the Oracle Database Express Edition interface. The SQL command window contains the CREATE TABLE statement for AdminRoles. The message 'Table created.' is shown in the results pane, along with '0.02 seconds'.

Index: CREATE INDEX adminroles\_idx ON AdminRoles(a\_id, r\_id);

<a href="#">Results</a>	<a href="#">Explain</a>	<a href="#">Describe</a>	<a href="#">Saved SQL</a>	<a href="#">History</a>
Index created.				

0.02 seconds

### Table: Agency Staff

Sequence: CREATE SEQUENCE staff\_seq START WITH 80001 INCREMENT BY 1 NOCACHE NOCYCLE;

Table Query:

```
CREATE TABLE Agency_Staff (
    s_id NUMBER(10) PRIMARY KEY,
    s_name VARCHAR2(100),
    s_password VARCHAR2(255) NOT NULL,
    s_address_city VARCHAR2(100),
    s_address_street VARCHAR2(100),
    s_address_house VARCHAR2(100),
    s_gender CHAR(1),
    s_email VARCHAR2(100) UNIQUE,
    s_salary DECIMAL(10, 2),
    s_role VARCHAR2(50),
    s_age NUMBER(3)
);
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command entered is:

```
CREATE TABLE Agency_Staff (
    s_id NUMBER(10) PRIMARY KEY,
    s_name VARCHAR2(100),
    s_address_city VARCHAR2(100),
    s_address_street VARCHAR2(100),
    s_address_house VARCHAR2(100),
    s_gender CHAR(1),
    s_email VARCHAR2(100),
    s_salary DECIMAL(10, 2),
    s_role VARCHAR2(50),
    s_age NUMBER(3)
);
```

The results pane shows the output:

```
Table created.
0.00 seconds
```

Application Express 2.1.0.0.39
Copyright © 1999, 2006, Oracle. All rights reserved.

Index: CREATE INDEX staff\_idx\_email ON AgencyStaff(s\_id, s\_email);

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The results pane shows the output:

```
Index created.
0.00 seconds
```

### Table: Reports

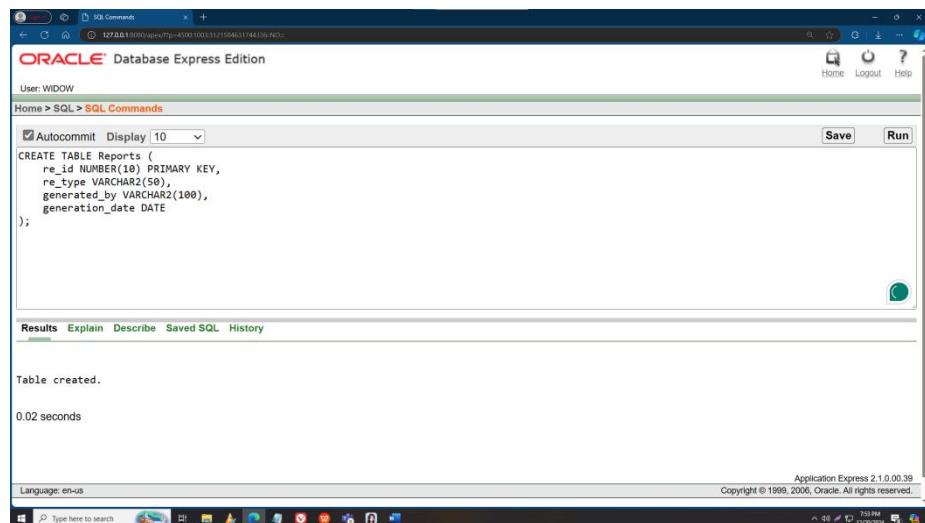
Sequence: CREATE SEQUENCE reports\_seq START WITH 90001 INCREMENT BY 1 NOCACHE NOCYCLE;

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The results pane shows the output:

```
Sequence created.
0.00 seconds
```

Table Query:

```
CREATE TABLE Reports (
    re_id NUMBER(10) PRIMARY KEY,
    re_type VARCHAR2(50),
    generated_by VARCHAR2(100),
    generation_date DATE
);
```



The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the following SQL code is entered:

```
CREATE TABLE Reports (
    re_id NUMBER(10) PRIMARY KEY,
    re_type VARCHAR2(50),
    generated_by VARCHAR2(100),
    generation_date DATE
);
```

Below the code, the results show:

Table created.  
0.02 seconds

At the bottom, the status bar indicates "Language: en-us" and "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved."

Table Index:



The screenshot shows the Oracle Database Express Edition interface. The results section displays:

Index created.  
0.00 seconds

### Table: Staff Reports

Table Query:

```
CREATE TABLE StaffReports (
    s_id NUMBER(10),
    re_id NUMBER(10),
    PRIMARY KEY (s_id, re_id),
```

```

FOREIGN KEY (s_id) REFERENCES AgencyStaff(s_id),
FOREIGN KEY (re_id) REFERENCES Reports(re_id)
);

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL editor contains the following code:

```

CREATE TABLE StaffReports (
    s_id NUMBER(10),
    re_id NUMBER(10),
    PRIMARY KEY (s_id, re_id),
    FOREIGN KEY (s_id) REFERENCES AgencyStaff(s_id),
    FOREIGN KEY (re_id) REFERENCES Reports(re_id)
);

```

The results pane below the editor shows the output:

```

Table created.

0.02 seconds

```

The status bar at the bottom indicates "Language: en-us" and "Copyright © 1999, 2006, Oracle. All rights reserved.".

Index: CREATE INDEX staffreports\_idx ON StaffReports(s\_id, re\_id);

The screenshot shows the Application Express 2.1.0.0.39 interface. The SQL editor contains the following code:

```

CREATE INDEX staffreports_idx ON StaffReports(s_id, re_id);

```

The results pane below the editor shows the output:

```

Index created.

0.00 seconds

```

### Table: Stuff-Admin

#### Sequence:

```

CREATE SEQUENCE admin_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE staff_seq START WITH 1 INCREMENT BY 1;

```

The screenshot shows a SQL command window with the following output:

```
Results Explain Describe Saved SQL History
```

Sequence created.  
0.02 seconds

## Table Creation:

```
CREATE TABLE Hires (
    a_id NUMBER(10),
    s_id NUMBER(10),
    PRIMARY KEY (a_id, s_id),
    FOREIGN KEY (a_id) REFERENCES Admin(a_id),
    FOREIGN KEY (s_id) REFERENCES AgencyStaff(s_id)
);
```

The screenshot shows a SQL command window in Oracle Database Express Edition with the following output:

```
Home > SQL > SQL Commands
```

```
CREATE TABLE Hires (
    a_id NUMBER(10),
    s_id NUMBER(10),
    PRIMARY KEY (a_id, s_id),
    FOREIGN KEY (a_id) REFERENCES Admin(a_id),
    FOREIGN KEY (s_id) REFERENCES AgencyStaff(s_id)
);
```

Table created.  
0.00 seconds

Language: en-us

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## Data Insertion

**Table: Homeowners**

```
INSERT INTO Homeowners (ho_id, ho_name, ho_email, ho_password, ho_address_city,  
ho_address_street, ho_address_house, ho_phone, ho_gender, ho_age)
```

```
VALUES (1, 'John Doe', 'john.doe@example.com', 'pass123', 'New York', '5th Avenue', 'Apt  
12B', '1234567890', 'M', 35);
```

```
INSERT INTO Homeowners (ho_id, ho_name, ho_email, ho_password, ho_address_city,  
ho_address_street, ho_address_house, ho_phone, ho_gender, ho_age)
```

```
VALUES (2, 'Jane Smith', 'jane.smith@example.com', 'pass123', 'Los Angeles', 'Sunset Blvd',  
'House 24', '2345678901', 'F', 29);
```

```
INSERT INTO Homeowners (ho_id, ho_name, ho_email, ho_password, ho_address_city,  
ho_address_street, ho_address_house, ho_phone, ho_gender, ho_age)
```

```
VALUES (3, 'Robert Brown', 'robert.brown@example.com', 'pass123', 'Chicago', 'Lake Shore  
Dr', 'Unit 7C', '3456789012', 'M', 42);
```

```
INSERT INTO Homeowners (ho_id, ho_name, ho_email, ho_password, ho_address_city,  
ho_address_street, ho_address_house, ho_phone, ho_gender, ho_age)
```

```
VALUES (4, 'Emily Johnson', 'emily.johnson@example.com', 'pass123', 'Houston', 'Main St',  
'Suite 10A', '4567890123', 'F', 31);
```

```
INSERT INTO Homeowners (ho_id, ho_name, ho_email, ho_password, ho_address_city,  
ho_address_street, ho_address_house, ho_phone, ho_gender, ho_age)
```

```
VALUES (5, 'Michael Davis', 'michael.davis@example.com', 'pass123', 'San Francisco',  
'Market St', 'Loft 5D', '5678901234', 'M', 38);
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The user is connected as 'WIDOW'. The SQL command entered is 'select \* from homeowners'. The results are displayed in a grid:

HO_ID	HO_NAME	HO_EMAIL	HO_ADDRESS_CITY	HO_ADDRESS_STREET	HO_ADDRESS_HOUSE	HO_PHONE	HO_GENDER	HO AGE
1	John Doe	john.doe@example.com	New York	5th Ave	12A	1234567890	M	45
2	Jane Smith	jane.smith@example.com	Los Angeles	Sunset Blvd	20B	0987654321	F	35
3	Alice Johnson	alice.johnson@example.com	Chicago	Michigan Ave	15C	1122334455	F	28
4	Bob Brown	bob.brown@example.com	Houston	Main St	8D	2233445566	M	50
5	Charlie Davis	charlie.davis@example.com	San Francisco	Market St	5E	3344556677	M	40

5 rows returned in 0.02 seconds [CSV Export](#)

Language: en-us Copyright © 1999, 2006, Oracle. All rights reserved.

## Table: Jobs

```
INSERT INTO Jobs (job_id, ho_id, job_title, status, schedule, postdate) VALUES (1, 1, 'Lawn Mowing', 'Completed', TO_DATE('2024-04-10', 'YYYY-MM-DD'), TO_DATE('2024-04-01', 'YYYY-MM-DD'));
```

```
INSERT INTO Jobs (job_id, ho_id, job_title, status, schedule, postdate) VALUES (2, 2, 'Roof Repair', 'Pending', TO_DATE('2024-04-20', 'YYYY-MM-DD'), TO_DATE('2024-04-10', 'YYYY-MM-DD'));
```

```
INSERT INTO Jobs (job_id, ho_id, job_title, status, schedule, postdate) VALUES (3, 3, 'Painting', 'In Progress', TO_DATE('2024-04-15', 'YYYY-MM-DD'), TO_DATE('2024-04-05', 'YYYY-MM-DD'));
```

```
INSERT INTO Jobs (job_id, ho_id, job_title, status, schedule, postdate) VALUES (4, 4, 'Electrical Fix', 'Completed', TO_DATE('2024-04-12', 'YYYY-MM-DD'), TO_DATE('2024-04-02', 'YYYY-MM-DD'));
```

```
INSERT INTO Jobs (job_id, ho_id, job_title, status, schedule, postdate) VALUES (5, 5, 'Plumbing', 'Pending', TO_DATE('2024-04-25', 'YYYY-MM-DD'), TO_DATE('2024-04-15', 'YYYY-MM-DD'));
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The user is running a query to select all columns from the JOBS table. The results are displayed in a grid format:

JOB_ID	HO_ID	JOB_TITLE	STATUS	SCHEDULE	POSTDATE
1	1	Lawn Mowing	Completed	10-APR-24	01-APR-24
2	2	Roof Repair	Pending	20-APR-24	10-APR-24
3	3	Painting	In Progress	15-APR-24	05-APR-24
4	4	Electrical Fix	Completed	12-APR-24	02-APR-24
5	5	Plumbing	Pending	25-APR-24	15-APR-24

5 rows returned in 0.00 seconds

CSV Export

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

**Table: Homeowner - Jobs**

```
INSERT INTO HomeownerJobs (ho_id, job_id) VALUES (1, 1);
INSERT INTO HomeownerJobs (ho_id, job_id) VALUES (2, 2);
INSERT INTO HomeownerJobs (ho_id, job_id) VALUES (3, 3);
INSERT INTO HomeownerJobs (ho_id, job_id) VALUES (4, 4);
INSERT INTO HomeownerJobs (ho_id, job_id) VALUES (5, 5);
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The user is running a query to select all columns from the homeownerjobs table. The results are displayed in a grid format:

HO_ID	JOB_ID
1	1
2	2
3	3
4	4
5	5

5 rows returned in 0.00 seconds

CSV Export

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## Table: Payments

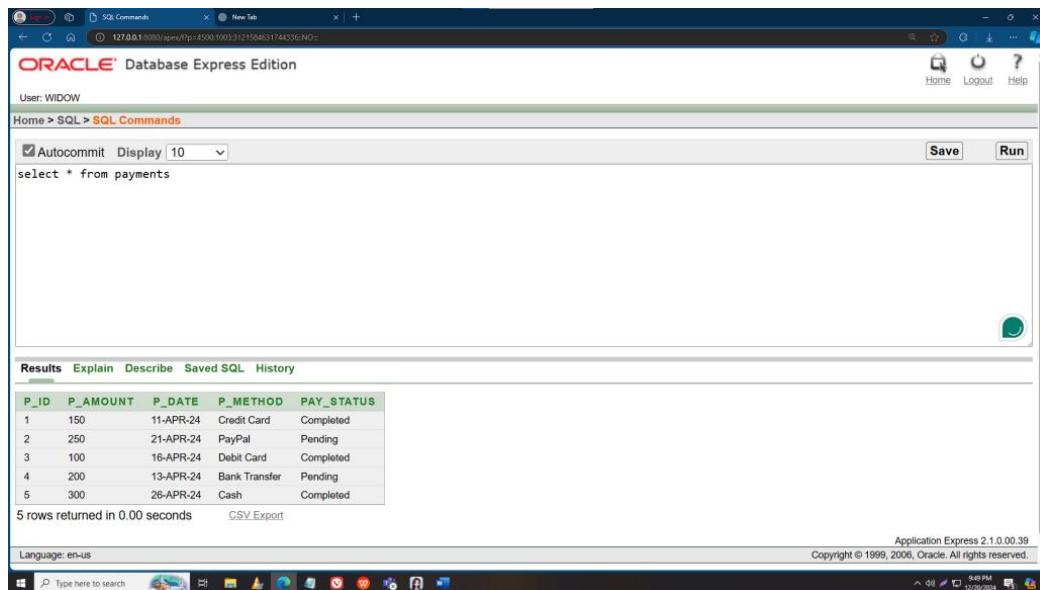
```
INSERT INTO Payments (p_id, p_amount, p_date, p_method, pay_status) VALUES (1, 150.00, TO_DATE('2024-04-11', 'YYYY-MM-DD'), 'Credit Card', 'Completed');
```

```
INSERT INTO Payments (p_id, p_amount, p_date, p_method, pay_status) VALUES (2, 250.00, TO_DATE('2024-04-21', 'YYYY-MM-DD'), 'PayPal', 'Pending');
```

```
INSERT INTO Payments (p_id, p_amount, p_date, p_method, pay_status) VALUES (3, 100.00, TO_DATE('2024-04-16', 'YYYY-MM-DD'), 'Debit Card', 'Completed');
```

```
INSERT INTO Payments (p_id, p_amount, p_date, p_method, pay_status) VALUES (4, 200.00, TO_DATE('2024-04-13', 'YYYY-MM-DD'), 'Bank Transfer', 'Pending');
```

```
INSERT INTO Payments (p_id, p_amount, p_date, p_method, pay_status) VALUES (5, 300.00, TO_DATE('2024-04-26', 'YYYY-MM-DD'), 'Cash', 'Completed');
```



The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands tab, a query is run:

```
select * from payments
```

The results are displayed in a grid:

P_ID	P_AMOUNT	P_DATE	P_METHOD	PAY_STATUS
1	150	11-APR-24	Credit Card	Completed
2	250	21-APR-24	PayPal	Pending
3	100	16-APR-24	Debit Card	Completed
4	200	13-APR-24	Bank Transfer	Pending
5	300	26-APR-24	Cash	Completed

5 rows returned in 0.00 seconds

CSV Export

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## Table: Job-Payments

```
INSERT INTO JobPayments (job_id, p_id) VALUES (1, 1);
```

```
INSERT INTO JobPayments (job_id, p_id) VALUES (2, 2);
```

```
INSERT INTO JobPayments (job_id, p_id) VALUES (3, 3);
```

```

INSERT INTO JobPayments (job_id, p_id) VALUES (4, 4);
INSERT INTO JobPayments (job_id, p_id) VALUES (5, 5);

```

The screenshot shows the Oracle Database Express Edition interface. The title bar says "SQL Commands". The main area contains the following SQL code:

```
select * from jobpayments
```

Below the code, the results are displayed in a table:

JOB_ID	P_ID
1	1
2	2
3	3
4	4
5	5

Text at the bottom of the results pane: "5 rows returned in 0.00 seconds" and "CSV Export".

At the bottom right of the window: "Application Express 2.1.0.0.39" and "Copyright © 1999, 2006, Oracle. All rights reserved."

### Table: Helpers

```

INSERT INTO Helpers (he_id, he_name, he_password, he_address_city, he_address_street,
he_address_house, he_gender, he_phone)
VALUES (1, 'Alice Green', 'pass123', 'New York', 'Madison Ave', 'Apt 3C', 'F', '1234567890');

INSERT INTO Helpers (he_id, he_name, he_password, he_address_city, he_address_street,
he_address_house, he_gender, he_phone)
VALUES (2, 'Brian Adams', 'pass123', 'Los Angeles', 'Hollywood Blvd', 'House 14', 'M',
'2345678901');

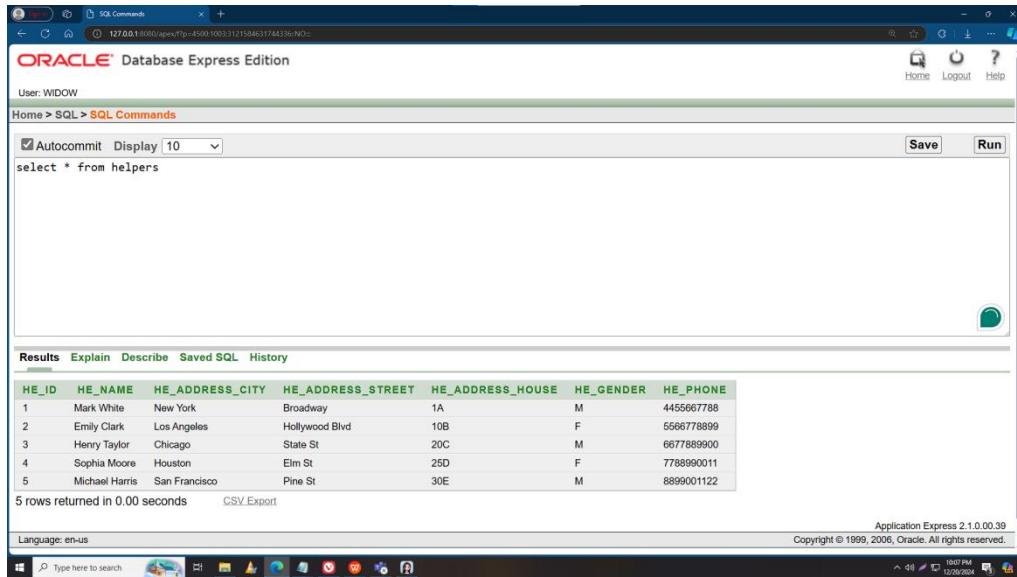
INSERT INTO Helpers (he_id, he_name, he_password, he_address_city, he_address_street,
he_address_house, he_gender, he_phone)
VALUES (3, 'Sophia Carter', 'pass123', 'Chicago', 'Michigan Ave', 'Unit 5B', 'F',
'3456789012');

INSERT INTO Helpers (he_id, he_name, he_password, he_address_city, he_address_street,
he_address_house, he_gender, he_phone)
VALUES (4, 'Daniel White', 'pass123', 'Houston', 'Broadway St', 'Suite 22A', 'M',
'4567890123');

```

```
INSERT INTO Helpers (he_id, he_name, he_password, he_address_city, he_address_street,  
he_address_house, he_gender, he_phone)
```

```
VALUES (5, 'Emma Wilson', 'pass123', 'San Francisco', 'Van Ness Ave', 'Loft 7D', 'F',  
'5678901234');
```



The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command entered is "select \* from helpers". The results are displayed in a grid:

HE_ID	HE_NAME	HE_ADDRESS_CITY	HE_ADDRESS_STREET	HE_ADDRESS_HOUSE	HE_GENDER	HE_PHONE
1	Mark White	New York	Broadway	1A	M	4455667788
2	Emily Clark	Los Angeles	Hollywood Blvd	10B	F	5566778899
3	Henry Taylor	Chicago	State St	20C	M	6677889900
4	Sophia Moore	Houston	Elm St	25D	F	7788990011
5	Michael Harris	San Francisco	Pine St	30E	M	8899001122

Below the table, it says "5 rows returned in 0.00 seconds". The bottom status bar shows "Application Express 2.1.0.0.0.39" and "Copyright © 1999, 2006, Oracle. All rights reserved."

**Table: Helper-Availability**

```
INSERT INTO Helper_Availability (he_id, availability) VALUES (1, 'Morning');
```

```
INSERT INTO Helper_Availability (he_id, availability) VALUES (2, 'Afternoon');
```

```
INSERT INTO Helper_Availability (he_id, availability) VALUES (3, 'Evening');
```

```
INSERT INTO Helper_Availability (he_id, availability) VALUES (4, 'Morning');
```

```
INSERT INTO Helper_Availability (he_id, availability) VALUES (5, 'Afternoon');
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. A SQL command `select \* from Helper\_Availability` is run, resulting in the following output:

HE_ID	AVAILABILITY
1	Morning
2	Afternoon
3	Evening
4	Morning
5	Afternoon

5 rows returned in 0.00 seconds

## Table: Feedbacks

```
INSERT INTO Feedbacks (f_id, rating, comments, apply_date, f_date) VALUES (1, 5, 'Excellent job!', TO_DATE('2024-04-10', 'YYYY-MM-DD'), TO_DATE('2024-04-11', 'YYYY-MM-DD'));
```

```
INSERT INTO Feedbacks (f_id, rating, comments, apply_date, f_date) VALUES (2, 4, 'Good work!', TO_DATE('2024-04-20', 'YYYY-MM-DD'), TO_DATE('2024-04-21', 'YYYY-MM-DD'));
```

```
INSERT INTO Feedbacks (f_id, rating, comments, apply_date, f_date) VALUES (3, 3, 'Satisfactory', TO_DATE('2024-04-15', 'YYYY-MM-DD'), TO_DATE('2024-04-16', 'YYYY-MM-DD'));
```

```
INSERT INTO Feedbacks (f_id, rating, comments, apply_date, f_date) VALUES (4, 2, 'Needs improvement', TO_DATE('2024-04-12', 'YYYY-MM-DD'), TO_DATE('2024-04-13', 'YYYY-MM-DD'));
```

```
INSERT INTO Feedbacks (f_id, rating, comments, apply_date, f_date) VALUES (5, 1, 'Poor service', TO_DATE('2024-04-25', 'YYYY-MM-DD'), TO_DATE('2024-04-26', 'YYYY-MM-DD'));
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. A user named 'WIDOW' is logged in. The SQL command entered is 'select \* from feedbacks'. The results section displays a table with 5 rows:

F_ID	RATING	COMMENTS	APPLY_DATE	F_DATE
1	5	Excellent job!	10-APR-24	11-APR-24
2	4	Good work!	20-APR-24	21-APR-24
3	3	Satisfactory	15-APR-24	16-APR-24
4	2	Needs improvement	12-APR-24	13-APR-24
5	1	Poor service	25-APR-24	26-APR-24

5 rows returned in 0.00 seconds

**Table: Homeowner-Feedbacks**

```

INSERT INTO HomeownerFeedbacks (ho_id, f_id) VALUES (1, 1);
INSERT INTO HomeownerFeedbacks (ho_id, f_id) VALUES (2, 2);
INSERT INTO HomeownerFeedbacks (ho_id, f_id) VALUES (3, 3);
INSERT INTO HomeownerFeedbacks (ho_id, f_id) VALUES (4, 4);
INSERT INTO HomeownerFeedbacks (ho_id, f_id) VALUES (5, 5);

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. A user named 'WIDOW' is logged in. The SQL command entered is 'select \* from homeownerfeedbacks'. The results section displays a table with 5 rows:

HO_ID	F_ID
1	1
2	2
3	3
4	4
5	5

5 rows returned in 0.00 seconds

**Table: Helper-Feedbacks**

```
INSERT INTO HelperFeedbacks (he_id, f_id) VALUES (1, 1);
INSERT INTO HelperFeedbacks (he_id, f_id) VALUES (2, 2);
INSERT INTO HelperFeedbacks (he_id, f_id) VALUES (3, 3);
INSERT INTO HelperFeedbacks (he_id, f_id) VALUES (4, 4);
INSERT INTO HelperFeedbacks (he_id, f_id) VALUES (5, 5);
```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a query is run:

```
select * from helperfeedbacks where 1=1
```

The results pane displays the following data:

HE_ID	F_ID
1	1
2	2
3	3
4	4
5	5

5 rows returned in 0.00 seconds

**Table: Roles**

```
INSERT INTO Roles (r_id, r_name, r_permission) VALUES (1, 'Manager', 'Full');
INSERT INTO Roles (r_id, r_name, r_permission) VALUES (2, 'Assistant', 'Partial');
INSERT INTO Roles (r_id, r_name, r_permission) VALUES (3, 'Clerk', 'Read-Only');
INSERT INTO Roles (r_id, r_name, r_permission) VALUES (4, 'Supervisor', 'Full');
INSERT INTO Roles (r_id, r_name, r_permission) VALUES (5, 'Admin', 'Full');
```

\

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command entered is "select \* from roles". The results show four rows of data:

R_ID	R_NAME	R_PERMISSION
1	Manager	Full
2	Assistant	Partial
3	Clerk	Read-Only
4	Admin	Full
5	Supervisor	Full

5 rows returned in 0.00 seconds

**Table: Admins**

```

INSERT INTO Admins (a_id, a_name, a_password, a_address_city, a_address_street,
a_address_house, a_gender, a_phone, a_email, a_age)

VALUES (1, 'Oliver King', 'pass123', 'New York', 'Lexington Ave', 'Apt 10A', 'M',
'1234509876', 'oliver.king@example.com', 40);

INSERT INTO Admins (a_id, a_name, a_password, a_address_city, a_address_street,
a_address_house, a_gender, a_phone, a_email, a_age)

VALUES (2, 'Sophia Martinez', 'pass123', 'Los Angeles', 'Beverly Blvd', 'House 27', 'F',
'2345608765', 'sophia.martinez@example.com', 35);

INSERT INTO Admins (a_id, a_name, a_password, a_address_city, a_address_street,
a_address_house, a_gender, a_phone, a_email, a_age)

VALUES (3, 'Ethan Wright', 'pass123', 'Chicago', 'State St', 'Unit 9C', 'M', '3456707654',
'ethan.wright@example.com', 45);

INSERT INTO Admins (a_id, a_name, a_password, a_address_city, a_address_street,
a_address_house, a_gender, a_phone, a_email, a_age)

VALUES (4, 'Ava Johnson', 'pass123', 'Houston', 'Westheimer Rd', 'Suite 18B', 'F',
'4567806543', 'ava.johnson@example.com', 38);

INSERT INTO Admins (a_id, a_name, a_password, a_address_city, a_address_street,
a_address_house, a_gender, a_phone, a_email, a_age)

VALUES (5, 'William Brown', 'pass123', 'San Francisco', 'Mission St', 'Loft 3D', 'M',
'5678905432', 'william.brown@example.com', 50);

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. In the SQL editor, the following query is entered:

```
select * from Admins
```

The results pane displays a table with the following data:

A_ID	A_NAME	A_ADDRESS_CITY	A_ADDRESS_STREET	A_ADDRESS_HOUSE	A_GENDER	A_PHONE	A_EMAIL	A_AGE
1	James Brown	New York	5th Ave	10A	M	9900112233	james.brown@example.com	50
2	Anna Black	Los Angeles	Sunset Blvd	12B	F	8800112233	anna.black@example.com	45
3	Robert Green	Chicago	Michigan Ave	8C	M	7700112233	robert.green@example.com	55
4	Laura White	Houston	Main St	6D	F	6600112233	laura.white@example.com	40
5	Kevin Blue	San Francisco	Market St	3E	M	5500112233	kevin.blue@example.com	35

5 rows returned in 0.00 seconds

CSV Export

Application Express 2.1.0.0.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## Tables: Admin-Roles

```
INSERT INTO AdminRoles (a_id, r_id) VALUES (1, 1);
INSERT INTO AdminRoles (a_id, r_id) VALUES (2, 2);
INSERT INTO AdminRoles (a_id, r_id) VALUES (3, 3);
INSERT INTO AdminRoles (a_id, r_id) VALUES (4, 4);
INSERT INTO AdminRoles (a_id, r_id) VALUES (5, 5);
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. In the SQL editor, the following query is entered:

```
select * from adminroles
```

The results pane displays a table with the following data:

A_ID	R_ID
1	1
2	2
3	3
4	4
5	5

5 rows returned in 0.00 seconds

CSV Export

Application Express 2.1.0.0.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## Tables: Agency-Staff

```
INSERT INTO Agency_Staff (s_id, s_name, s_password, s_address_city, s_address_street,
s_address_house, s_gender, s_email, s_salary, s_role, s_age)
```

```
VALUES (1, 'Liam Carter', 'pass123', 'New York', 'Wall Street', 'Apt 8A', 'M',
'liam.carter@example.com', 55000.00, 'Manager', 40);
```

```
INSERT INTO Agency_Staff (s_id, s_name, s_password, s_address_city, s_address_street,
s_address_house, s_gender, s_email, s_salary, s_role, s_age)
```

```
VALUES (2, 'Emma Green', 'pass123', 'Los Angeles', 'Sunset Blvd', 'House 32', 'F',
'emma.green@example.com', 48000.00, 'Coordinator', 35);
```

```
INSERT INTO Agency_Staff (s_id, s_name, s_password, s_address_city, s_address_street,
s_address_house, s_gender, s_email, s_salary, s_role, s_age)
```

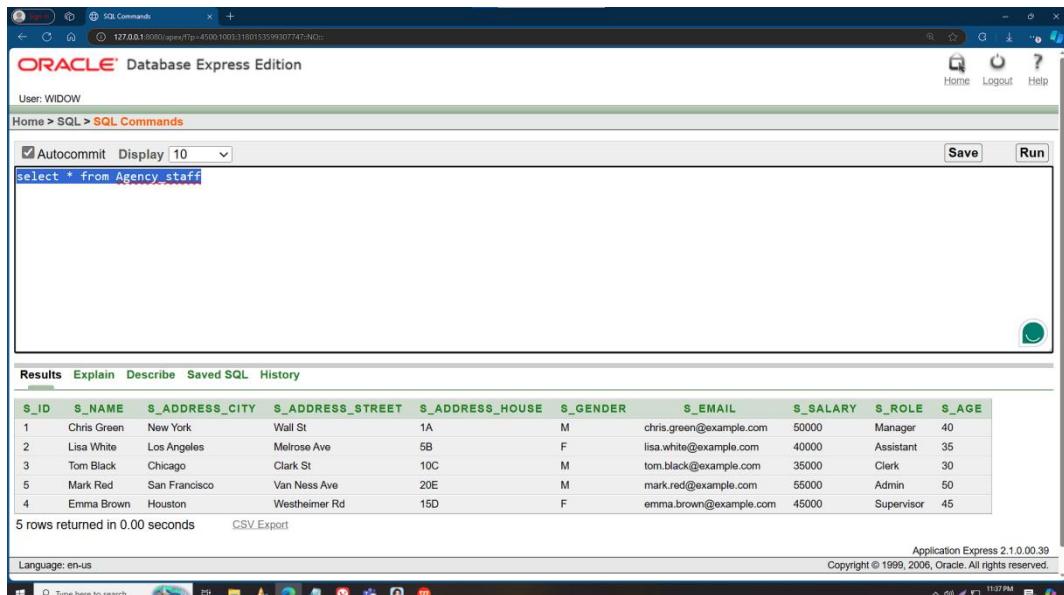
```
VALUES (3, 'Noah Mitchell', 'pass123', 'Chicago', 'Lake Shore Dr', 'Unit 11C', 'M',
'noah.mitchell@example.com', 60000.00, 'Supervisor', 45);
```

```
INSERT INTO Agency_Staff (s_id, s_name, s_password, s_address_city, s_address_street,
s_address_house, s_gender, s_email, s_salary, s_role, s_age)
```

```
VALUES (4, 'Olivia Adams', 'pass123', 'Houston', 'Main St', 'Suite 5B', 'F',
'olivia.adams@example.com', 52000.00, 'HR Specialist', 38);
```

```
INSERT INTO Agency_Staff (s_id, s_name, s_password, s_address_city, s_address_street,
s_address_house, s_gender, s_email, s_salary, s_role, s_age)
```

```
VALUES (5, 'William Scott', 'pass123', 'San Francisco', 'Broadway St', 'Loft 2D', 'M',
'william.scott@example.com', 58000.00, 'Accountant', 50);
```



The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command entered is:

```
select * from Agency_Staff
```

The results section displays the following data:

S_ID	S_NAME	S_ADDRESS_CITY	S_ADDRESS_STREET	S_ADDRESS_HOUSE	S_GENDER	S_EMAIL	S_SALARY	S_ROLE	S_AGE
1	Chris Green	New York	Wall St	1A	M	chris.green@example.com	50000	Manager	40
2	Lisa White	Los Angeles	Melrose Ave	5B	F	lisa.white@example.com	40000	Assistant	35
3	Tom Black	Chicago	Clark St	10C	M	tom.black@example.com	35000	Clerk	30
4	Mark Red	San Francisco	Van Ness Ave	20E	M	mark.red@example.com	55000	Admin	50
5	Emma Brown	Houston	Westheimer Rd	15D	F	emma.brown@example.com	45000	Supervisor	45

5 rows returned in 0.00 seconds

CSV Export

Application Express 2.1.0.0.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## Table: Reports

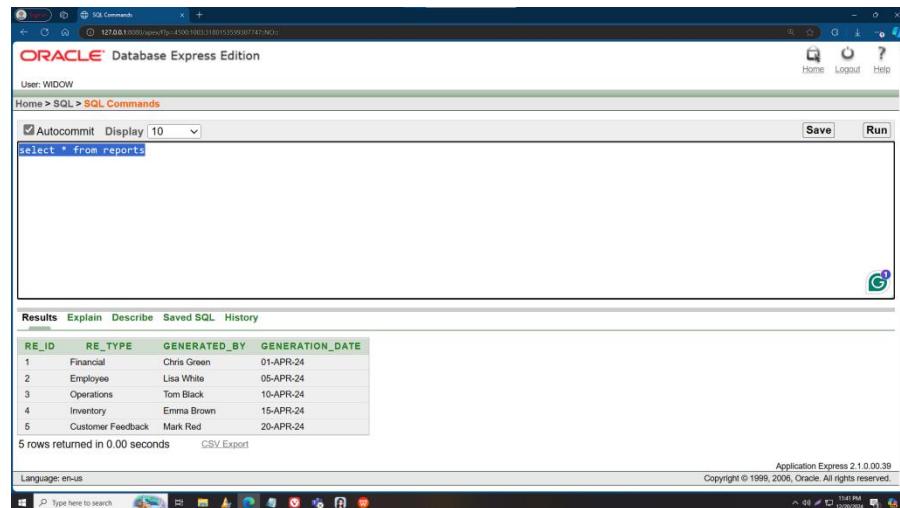
```
INSERT INTO Reports (re_id, re_type, generated_by, generation_date) VALUES (1, 'Financial', 'Chris Green', TO_DATE('2024-04-01', 'YYYY-MM-DD'));
```

```
INSERT INTO Reports (re_id, re_type, generated_by, generation_date) VALUES (2, 'Employee', 'Lisa White', TO_DATE('2024-04-05', 'YYYY-MM-DD'));
```

```
INSERT INTO Reports (re_id, re_type, generated_by, generation_date) VALUES (3, 'Operations', 'Tom Black', TO_DATE('2024-04-10', 'YYYY-MM-DD'));
```

```
INSERT INTO Reports (re_id, re_type, generated_by, generation_date) VALUES (4, 'Inventory', 'Emma Brown', TO_DATE('2024-04-15', 'YYYY-MM-DD'));
```

```
INSERT INTO Reports (re_id, re_type, generated_by, generation_date) VALUES (5, 'Customer Feedback', 'Mark Red', TO_DATE('2024-04-20', 'YYYY-MM-DD'));
```



The screenshot shows the Oracle Database Express Edition SQL Commands interface. A SQL command is entered in the text area:

```
select * from reports;
```

The results are displayed in a grid:

RE_ID	RE_TYPE	GENERATED_BY	GENERATION_DATE
1	Financial	Chris Green	01-APR-24
2	Employee	Lisa White	05-APR-24
3	Operations	Tom Black	10-APR-24
4	Inventory	Emma Brown	15-APR-24
5	Customer Feedback	Mark Red	20-APR-24

5 rows returned in 0.00 seconds

CSV Export

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## Reports: Staff Report

```
INSERT INTO StaffReports (s_id, re_id) VALUES (1, 1);
```

```
INSERT INTO StaffReports (s_id, re_id) VALUES (2, 2);
```

```
INSERT INTO StaffReports (s_id, re_id) VALUES (3, 3);
```

```
INSERT INTO StaffReports (s_id, re_id) VALUES (4, 4);
```

```
INSERT INTO StaffReports (s_id, re_id) VALUES (5, 5);
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. A user named 'WIDOW' is logged in. The query 'select \* from staffreports;' is run, resulting in the following table:

S_ID	RE_ID
1	1
2	2
3	3
4	4
5	5

5 rows returned in 0.00 seconds

CSV Export

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

**Table: Staff-Admin**

```
INSERT INTO Hires (a_id, s_id) VALUES (1, 101);
INSERT INTO Hires (a_id, s_id) VALUES (2, 102);
INSERT INTO Hires (a_id, s_id) VALUES (3, 103);
INSERT INTO Hires (a_id, s_id) VALUES (4, 104);
INSERT INTO Hires (a_id, s_id) VALUES (5, 105);
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. A user named 'WIDOW' is logged in. The query 'select \* from adminroles;' is run, resulting in the following table:

A_ID	R_ID
1	1
2	2
3	3
4	4
5	5

5 rows returned in 0.00 seconds

CSV Export

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## **Basic Query Writing**

### **● Exception Handling**

- 1. Write a PL/SQL block that retrieves a homeowner's email based on a given homeowner ID. If the ID does not exist, handle the NO\_DATA\_FOUND exception by displaying a message.**

```
DECLARE
    v_email VARCHAR2(100);
    v_ho_id NUMBER := 10; -- Assume this ID does not exist
BEGIN
    SELECT ho_email INTO v_email
    FROM Homeowners
    WHERE ho_id = v_ho_id;

    DBMS_OUTPUT.PUT_LINE('Homeowner Email: ' || v_email);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No homeowner found with ID ' ||
v_ho_id);
END;
/
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code entered is:

```

DECLARE
  v_email VARCHAR2(100);
  v_no_id NUMBER := 10; -- Assume this ID does not exist
BEGIN
  SELECT ho_email INTO v_email
  FROM Homeowners
  WHERE ho_id = v_no_id;
  DBMS_OUTPUT.PUT_LINE('Homeowner Email: ' || v_email);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No homeowner found with ID ' || v_no_id);
END;
/

```

The results pane shows the output:

```

Homeowner Email: ariana@example.com
Statement processed.

0.00 seconds

```

At the bottom right, it says "Application Express 2.1.0.0.39 Copyright © 1999, 2009, Oracle. All rights reserved."

## 2. Create a PL/SQL block that checks if a payment exists for a given job. If not, raise a user-defined exception.

```

DECLARE
  v_count NUMBER;
  v_job_id NUMBER := 6; -- Assume this job_id does not exist in JobPayments
  e_no_payment EXCEPTION;
BEGIN
  SELECT COUNT(*) INTO v_count
  FROM JobPayments
  WHERE job_id = v_job_id;

  IF v_count = 0 THEN
    RAISE e_no_payment;
  END IF;

  DBMS_OUTPUT.PUT_LINE('Payment found for Job ID ' || v_job_id);

```

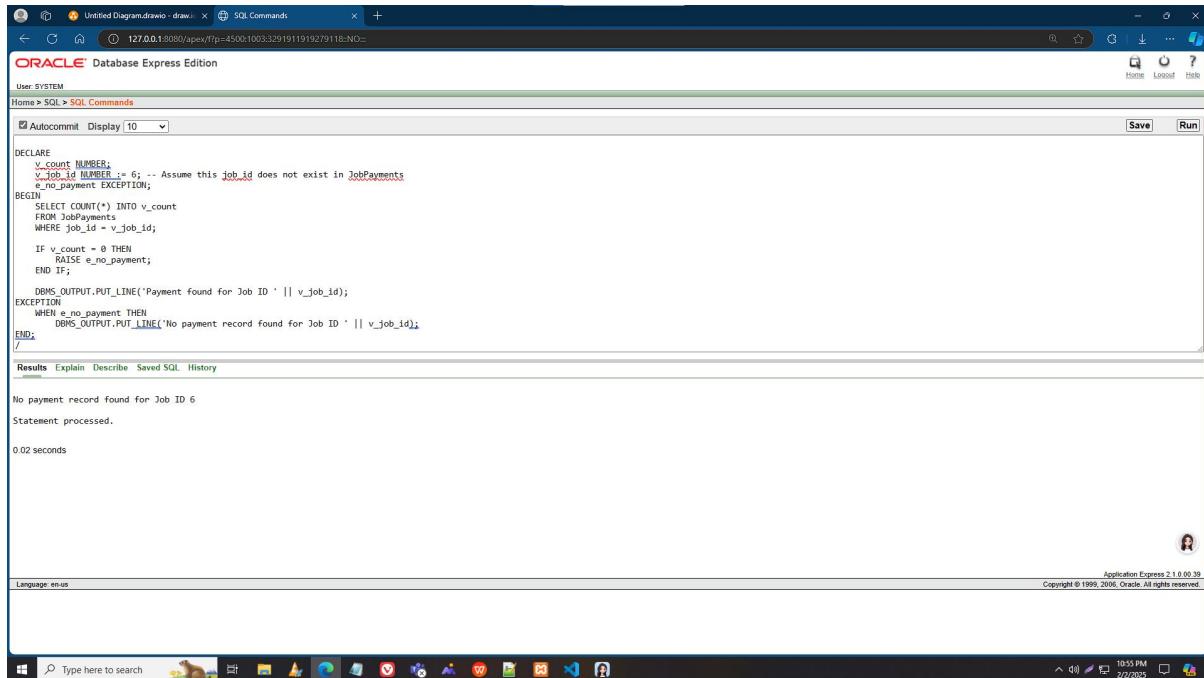
## EXCEPTION

```
WHEN e_no_payment THEN
```

```
    DBMS_OUTPUT.PUT_LINE('No payment record found for Job ID  
' || v_job_id);
```

```
END;
```

```
/
```



```
DECLARE
    v_count NUMBER;
    v_job_id NUMBER := 6; -- Assume this job_id does not exist in JobPayments
    e_no_payment EXCEPTION;
BEGIN
    SELECT COUNT(*) INTO v_count
    FROM JobPayments
    WHERE job_id = v_job_id;

    IF v_count = 0 THEN
        RAISE e_no_payment;
    END IF;

    DBMS_OUTPUT.PUT_LINE('Payment found for Job ID ' || v_job_id);
EXCEPTION
    WHEN e_no_payment THEN
        DBMS_OUTPUT.PUT_LINE('No payment record found for Job ID ' || v_job_id);
END;
/

```

No payment record found for Job ID 6  
Statement processed.  
0.02 seconds

Language: en-us

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## ● Implicit Locking

1. Write a query that selects a job record for update and prevents other sessions from modifying it.

```
SELECT *  
FROM Jobs  
WHERE job_id = 2  
FOR UPDATE;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command entered is:

```
SELECT *
FROM Jobs
WHERE job_id = 2
FOR UPDATE;
```

The results show one row:

JOB_ID	HO_ID	JOB_TITLE	STATUS	SCHEDULE	POSTDATE
2	2	Roof Repair	Pending	20-APR-24	10-APR-24

1 rows returned in 0.00 seconds

Language: en-us

Application Express 2.1.0.0.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

**2. Write a query that tries to lock a Jobs row but fails immediately if another session holds a lock.**

```
SELECT *
FROM Jobs
WHERE job_id = 2
FOR UPDATE NOWAIT
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command entered is:

```
SELECT *
FROM Jobs
WHERE job_id = 2
FOR UPDATE NOWAIT;
```

The results show one row:

JOB_ID	HO_ID	JOB_TITLE	STATUS	SCHEDULE	POSTDATE
2	2	Roof Repair	Pending	20-APR-24	10-APR-24

1 rows returned in 0.00 seconds

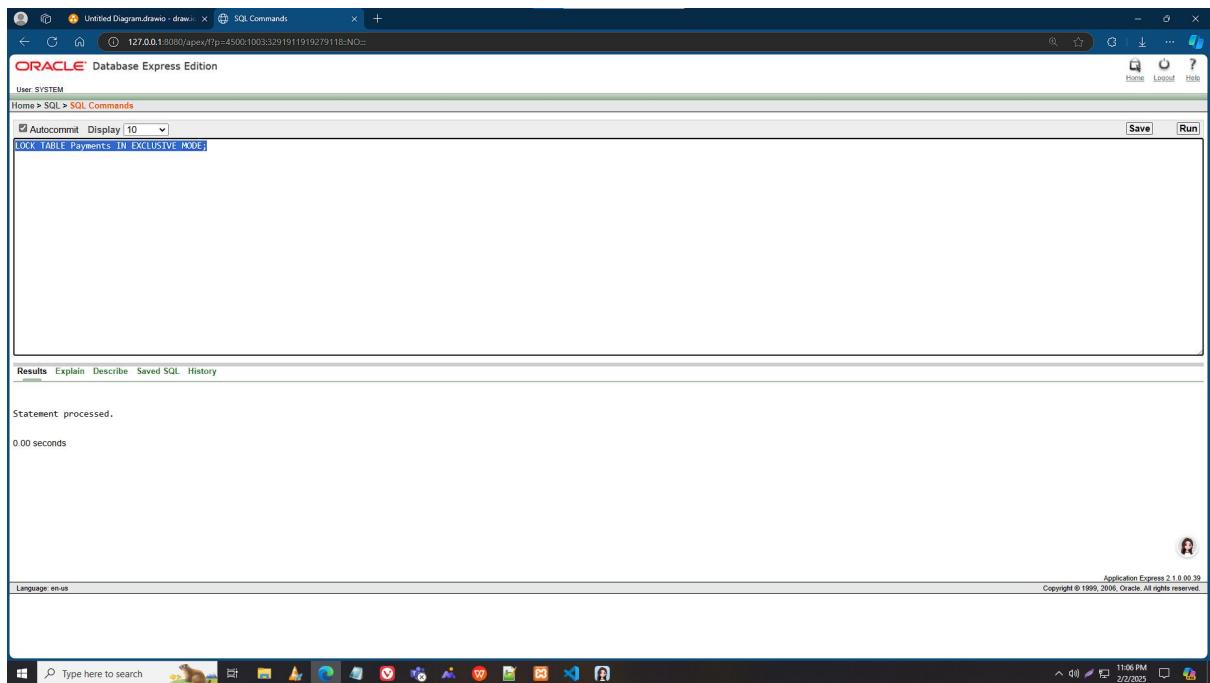
Language: en-us

Application Express 2.1.0.0.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## ● Explicit Locking

1. Write a query to lock the entire Payments table so only the current session can modify it.

LOCK TABLE Payments IN EXCLUSIVE MODE;



The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the following SQL command is entered:

```
LOCK TABLE Payments IN EXCLUSIVE MODE;
```

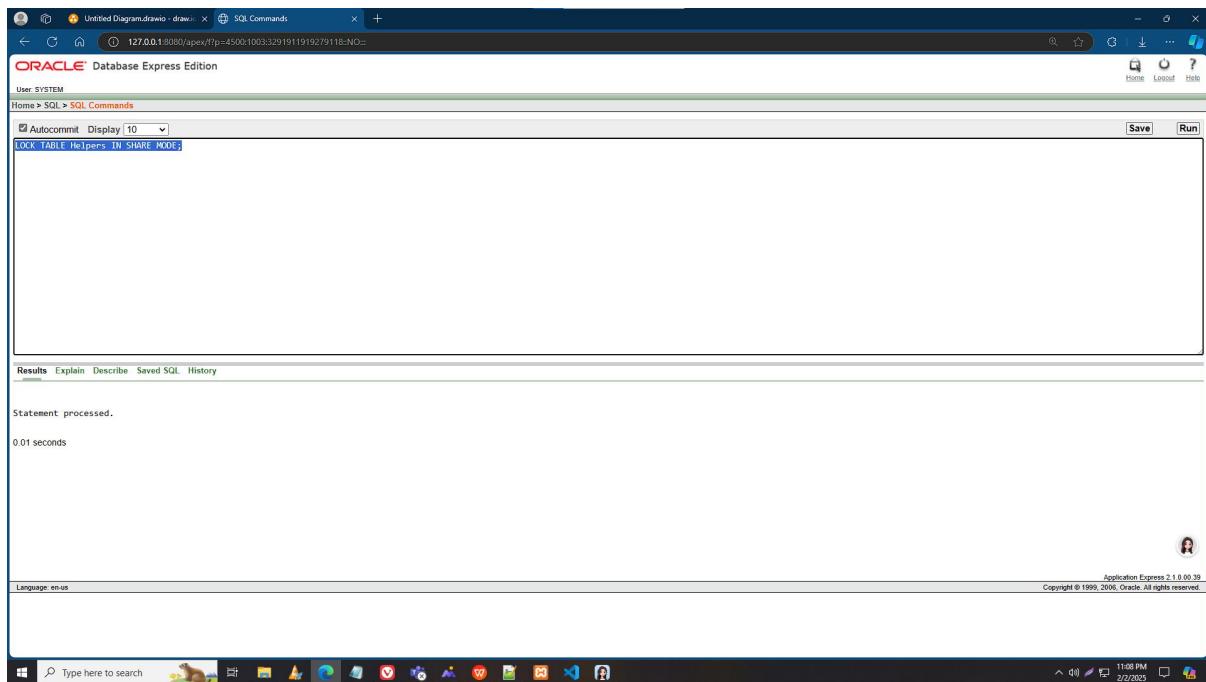
Below the command, the results show:

Statement processed.  
0.00 seconds

At the bottom right of the window, there is a note: "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved."

2. Write a query to allow only SELECT queries on the Helpers table but prevent updates or deletions.

LOCK TABLE Helpers IN SHARE MODE;



## ● Relational Algebra

**1. Express a relational algebra query to find all homeowners from New York.**

$$\sigma_{\text{ho\_address\_city}='NewYork'} (\text{Homeowners})$$

**2. Retrieve job titles where the job status is 'Pending'.**

$$\pi_{\text{job\_title}} (\sigma_{\text{status}='Pending'} (\text{Jobs}))$$

**3. Find homeowners who have both posted a job and made a payment.**

$$\pi_{\text{ho\_id}} (\text{Jobs}) \cap \pi_{\text{ho\_id}} (\text{JobPayments} \bowtie \text{Payments})$$

**4. Retrieve the names of homeowners who have provided feedback.**

$$\pi_{\text{ho\_name}} (\text{Homeowners} \bowtie \text{HomeownerFeedbacks})$$

**5. Find the total amount paid for each job using an aggregation operation.**

JobPayments $\bowtie$  Payments Gjob\_id, SUM(p\_amount) (JobPayments $\bowtie$  Payments)

**(Midterm Part)**

**● Variables**

**1. Write a PL/SQL block to retrieve the phone number of a specific homeowner (ho\_id = 2) and display it. Use a PL/SQL variable to store the phone number.**

```
DECLARE
    v_phone_number VARCHAR2(20);
    v_ho_id NUMBER := 2;
BEGIN
    SELECT ho_phone
    INTO v_phone_number
    FROM Homeowner
    WHERE ho_id = v_ho_id;

    DBMS_OUTPUT.PUT_LINE('The phone number for homeowner with ho_id ' ||
    v_ho_id || ' is: ' || v_phone_number);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No phone number found for homeowner with
        ho_id ' || v_ho_id);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```

DECLARE
  v_phone_number VARCHAR2(20);
  v_ho_id NUMBER := 2;
BEGIN
  SELECT ho_phone
  INTO v_phone_number
  FROM Homeowners
  WHERE ho_id = v_ho_id;

  DBMS_OUTPUT.PUT_LINE('The phone number for homeowner with ho_id ' || v_ho_id || ' is: ' || v_phone_number);
EXCEPTION
  WHEN NO_DATA_FOUND THEN

```

The results pane displays the output of the query:

```

The phone number for homeowner with ho_id 2 is: 0987654321
Statement processed.

0.02 seconds

```

At the bottom, it shows the application version and copyright information:

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

**2. Write a PL/SQL block to retrieve the job title for a specific job (job\_id = 1) and display it. Use a PL/SQL variable to store the job title.**

DECLARE

```
v_job_title VARCHAR2(100);    v_job_id NUMBER := 1;
```

BEGIN

```

SELECT job_title
INTO v_job_title
FROM Job
WHERE job_id = v_job_id;

```

```
DBMS_OUTPUT.PUT_LINE('The job title for job_id ' || v_job_id || ' is: ' || v_job_title);
```

EXCEPTION

```
WHEN NO_DATA_FOUND THEN
```

```

DBMS_OUTPUT.PUT_LINE('No job found for job_id ' || v_job_id);

WHEN OTHERS THEN

DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);

END;

```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a PL/SQL block is run. The code retrieves a job title for job\_id 1 and handles exceptions for no data found and other errors. The results show the job title 'Lawn Mowing' and a success message.

```

v_job_title VARCHAR2(100);  v_job_id NUMBER := 1;
BEGIN
  SELECT job_title
  INTO v_job_title
  FROM Job
  WHERE job_id = v_job_id;
  DBMS_OUTPUT.PUT_LINE('The job title for job_id ' || v_job_id || ' is: ' || v_job_title);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No job found for job_id ' || v_job_id);
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/

```

**Results:**

```

The job title for job_id 1 is: Lawn Mowing
Statement processed.

0.00 seconds

```

Language: en-us Application Express 2.1.0.0939 Copyright © 1999, 2006, Oracle. All rights reserved.

## ● Operators

1. Write a PL/SQL block that uses arithmetic operators to calculate the total payment for a specific homeowner. The total payment should be the sum of the payment amount and a 10% discount. Use variables to store the payment amount and the discount.

DECLARE

```

v_payment_amount NUMBER := 100.00;
v_discount NUMBER := 0.10;
v_total_payment NUMBER;

BEGIN
  v_total_payment := v_payment_amount + (v_payment_amount * v_discount);

```

```

DBMS_OUTPUT.PUT_LINE('Total payment after discount: ' || v_total_payment);

END;

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code in the editor window is:

```

DECLARE
    v_payment_amount NUMBER := 100.00;
    v_discount NUMBER := 0.10;
    v_total_payment NUMBER;
BEGIN
    v_total_payment := v_payment_amount + (v_payment_amount * v_discount);
    DBMS_OUTPUT.PUT_LINE('Total payment after discount: ' || v_total_payment);
END;
/

```

The results window shows the output:

```

Total payment after discount: 110
Statement processed.

0.00 seconds

```

At the bottom right of the interface, it says "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved."

**2. Write a PL/SQL block that uses logical operators to check if a homeowner's transaction (transaction\_id = 1) is above \$200 and the payment status is 'Paid'. Display a message indicating whether both conditions are true.**

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code in the editor window is:

```

DECLARE
    v_transaction_amount NUMBER := 250.00;
    v_payment_status VARCHAR2(20) := 'Paid';
BEGIN
    IF v_transaction_amount > 200 AND v_payment_status = 'Paid' THEN
        DBMS_OUTPUT.PUT_LINE('The transaction is above $200 and the payment status is Paid.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('The conditions are not met.');
    END IF;
END;
/

```

The results window shows the output:

```

The transaction is above $200 and the payment status is Paid.
Statement processed.

0.00 seconds

```

At the bottom right of the interface, it says "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved."

## ● Single\_Row Function

1. Write a PL/SQL block that uses the UPPER() function to convert the homeowner's name (ho\_name) for a given ho\_id = 101 to uppercase and displays the result.

```
DECLARE
    v_ho_name VARCHAR2(100);
BEGIN
    SELECT ho_name INTO v_ho_name
    FROM Homeowners
    WHERE ho_id = 1;
    DBMS_OUTPUT.PUT_LINE('Homeowner name in uppercase: ' || 
    UPPER(v_ho_name));
END;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL editor contains the following PL/SQL block:

```
DECLARE
    v_ho_name VARCHAR2(100);
BEGIN
    SELECT ho_name INTO v_ho_name
    FROM Homeowners
    WHERE ho_id = 1;
    DBMS_OUTPUT.PUT_LINE('Homeowner name in uppercase: ' || 
    UPPER(v_ho_name));
END;
```

The results pane shows the output of the block:

```
Homeowner name in uppercase: JOHN DOE
Statement processed.

0.00 seconds
```

At the bottom, the status bar indicates "Language: en-us" and "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved." The system tray at the bottom right shows the date and time as "12/21/2024 8:29 AM".

- 2. Write a PL/SQL block that uses the TO\_DATE() function to convert a string ('2023-08-01') into a date format and displays the result.**

```
DECLARE
```

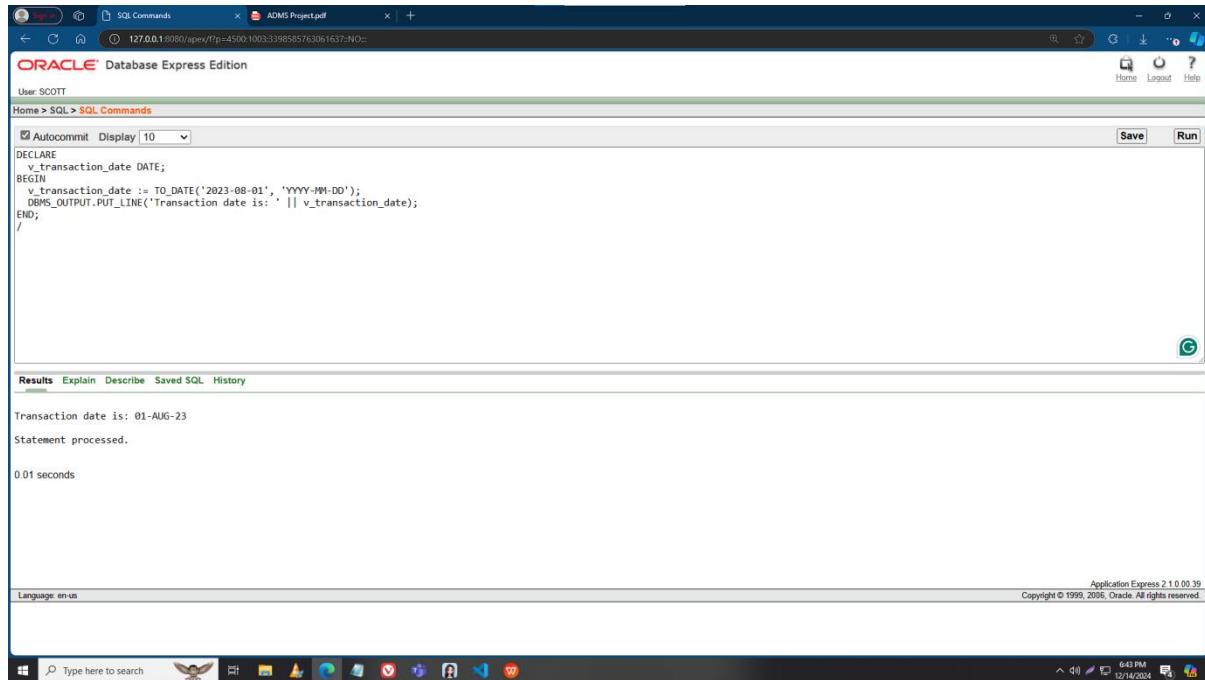
```
    v_transaction_date DATE;
```

```
BEGIN
```

```
    v_transaction_date := TO_DATE('2023-08-01', 'YYYY-MM-DD');
```

```
    DBMS_OUTPUT.PUT_LINE('Transaction date is: ' || v_transaction_date);
```

```
END;
```



```
ORACLE Database Express Edition
User SCOTT
Home > SQL > SQL Commands
Autocommit: Display: 10
Save Run
DECLARE
    v_transaction_date DATE;
BEGIN
    v_transaction_date := TO_DATE('2023-08-01', 'YYYY-MM-DD');
    DBMS_OUTPUT.PUT_LINE('Transaction date is: ' || v_transaction_date);
END;
/
Results Explain Describe Saved SQL History
Transaction date is: 01-AUG-23
Statement processed.
0.01 seconds
Language: en-us Application Express 2.1.0.0.39
Copyright © 1999, 2006, Oracle. All rights reserved.
643 PM 12/14/2024
```

## ● Group Function

- 1. Write a PL/SQL block that calculates the average rating from the Feedback table and displays the result.**

```
DECLARE
```

```
    v_avg_rating NUMBER;
```

```

BEGIN
    SELECT AVG(rating) INTO v_avg_rating
    FROM Feedbacks;

    DBMS_OUTPUT.PUT_LINE('Average rating: ' || v_avg_rating);

END;

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code entered is:

```

DECLARE
    v_avg_rating NUMBER;
BEGIN
    SELECT AVG(rating) INTO v_avg_rating
    FROM Feedbacks;

    DBMS_OUTPUT.PUT_LINE('Average rating: ' || v_avg_rating);
END;

```

The results pane shows the output:

```

Average rating: 3
Statement processed.

0.00 seconds

```

At the bottom, it says "Language: en-us" and "Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved."

## 2. Write a PL/SQL block that counts the number of Homeowner entries in the Homeowner table and displays the result.

```

DECLARE
    v_ho_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_ho_count
    FROM Homeowners;

```

```

DBMS_OUTPUT.PUT_LINE('Total number of homeowners: ' || v_ho_count);

END;

/

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```

DECLARE
v_ho_count NUMBER;
BEGIN
SELECT COUNT(*) INTO v_ho_count
FROM Homeowners;
DBMS_OUTPUT.PUT_LINE('Total number of homeowners: ' || v_ho_count);
END;
/

```

The results pane displays the output:

```

Total number of homeowners: 5
Statement processed.

0.00 seconds

```

At the bottom, it shows the language is en-us and the application version is Application Express 2.1.0.0.39.

## Loops

1. Write a PL/SQL block that uses a FOR loop to display the names of all admins from the Admins table who live in "New York".

```

DECLARE
    v_admin_name VARCHAR2(100);
BEGIN
    FOR rec IN (SELECT a_name FROM Admins WHERE a_address_city = 'New York')
    LOOP
        v_admin_name := rec.a_name;
        DBMS_OUTPUT.PUT_LINE('Admin Name: ' || v_admin_name);
    END LOOP;
END;

```



The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```

DECLARE
    v_admin_name VARCHAR2(100);
BEGIN
    FOR rec IN (SELECT a_name FROM Admins WHERE a_address_city = 'New York') LOOP
        v_admin_name := rec.a_name;
        DBMS_OUTPUT.PUT_LINE('Admin Name: ' || v_admin_name);
    END LOOP;
END;
/

```

The results pane displays the output of the PL/SQL block:

```

Admin Name: James Brown
Statement processed.

0.00 seconds

```

At the bottom right, it says "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved."

**2. Write a PL/SQL block that uses a WHILE loop to display the names of admins from the Admins table until an admin with an age greater than 50 is found.**

DECLARE

```

v_admin_name VARCHAR2(100);
v_admin_age NUMBER(3);
v_cursor SYS_REFCURSOR;

```

BEGIN

```

OPEN v_cursor FOR
    SELECT a_name, a_age
    FROM Admins
    ORDER BY a_id;

```

LOOP

```

    FETCH v_cursor INTO v_admin_name, v_admin_age;

```

```
    EXIT WHEN v_cursor%NOTFOUND OR v_admin_age > 50;
```

```
    DBMS_OUTPUT.PUT_LINE('Admin Name: ' || v_admin_name);  
END LOOP;
```

```
CLOSE v_cursor;
```

```
DBMS_OUTPUT.PUT_LINE('Loop terminated. Admin with age > 50 encountered or  
end of table reached.');
```

```
END;
```

```
/
```

The screenshot shows a Windows desktop environment with a dark theme. A browser window is open to the Oracle Application Express SQL Commands page at the URL `127.0.0.1:8080/apex/?p=45001003:3180153599307747:No..`. The page title is "SQL Commands". The SQL code entered is:

```
DECLARE  
  v_admin_name VARCHAR2(100);  
  v_admin_age NUMBER(3);  
  v_cursor SYS REFCURSOR;  
BEGIN  
  OPEN v_cursor FOR  
    SELECT a_name, a_age  
    FROM Admins  
    ORDER BY a_id;  
  IF v_admin_age > 50 THEN  
    DBMS_OUTPUT.PUT_LINE('Admin Name: ' || v_admin_name);  
  END IF;  
  EXIT WHEN v_cursor%NOTFOUND OR v_admin_age > 50;  
END LOOP;  
CLOSE v_cursor;  
DBMS_OUTPUT.PUT_LINE('Loop terminated. Admin with age > 50 encountered or end of table reached.');
```

The results section displays the output of the PL/SQL block:

```
Admin Name: James Brown  
Admin Name: Anna Black  
Loop terminated. Admin with age > 50 encountered or end of table reached.  
Statement processed.
```

At the bottom of the page, it says "0.00 seconds" and "Language: en-us". The footer includes the copyright notice "Copyright © 1999, 2006, Oracle. All rights reserved." and the version "Application Express 2.1.0.00.39".

## ● Conditional Statement

3. Write a PL/SQL block that uses a CASE statement to check the job\_title in the Job table. If the job\_title is 'Lawn Mowing', display "Outdoor Work". If the job\_title is 'House Cleaning', display "Indoor Work". Otherwise, display "Other Work".

```
DECLARE
    v_job_title VARCHAR2(100);
BEGIN
    SELECT job_title INTO v_job_title FROM Job WHERE job_id = 1;

    CASE
        WHEN v_job_title = 'Lawn Mowing' THEN
            DBMS_OUTPUT.PUT_LINE('Outdoor Work');
        WHEN v_job_title = 'House Cleaning' THEN
            DBMS_OUTPUT.PUT_LINE('Indoor Work');
        ELSE
            DBMS_OUTPUT.PUT_LINE('Other Work');
    END CASE;
END;
```

The screenshot shows the Oracle Database Express Edition SQL Commands window. The code entered is:

```

DECLARE
  v_job_title VARCHAR2(100);
BEGIN
  SELECT job_title INTO v_job_title FROM Job WHERE job_id = 1;
  CASE
    WHEN v_job_title = 'Lawn Mowing' THEN
      DBMS_OUTPUT.PUT_LINE('Outdoor Work');
    WHEN v_job_title = 'House Cleaning' THEN
      DBMS_OUTPUT.PUT_LINE('Indoor Work');
    ELSE
      DBMS_OUTPUT.PUT_LINE('Other Work');
  END CASE;
END;
/

```

The results pane shows the output:

```

Outdoor Work
Statement processed.

0.00 seconds

```

At the bottom right of the window, it says "Application Express 2.1.0.20.39 Copyright © 1999, 2006, Oracle. All rights reserved."

**1. Write a PL/SQL block that uses an IF-ELSE statement to check if a homeowner's ho\_phone number starts with "+1". If it does, display "US Number", otherwise display "International Number".**

DECLARE

v\_ho\_phone VARCHAR2(20);

BEGIN

SELECT ho\_phone INTO v\_ho\_phone FROM Homeowners WHERE ho\_id = 1;

IF v\_ho\_phone LIKE '+1%' THEN

DBMS\_OUTPUT.PUT\_LINE('US Number');

ELSE

DBMS\_OUTPUT.PUT\_LINE('International Number');

END IF;

END;

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```

DECLARE
    v_ho_phone VARCHAR2(20);
BEGIN
    SELECT ho_phone INTO v_ho_phone FROM Homeowners WHERE ho_id = 1;
    IF v_ho_phone LIKE '+1%' THEN
        DBMS_OUTPUT.PUT_LINE('US Number');
    ELSE
        DBMS_OUTPUT.PUT_LINE('International Number');
    END IF;
END;

```

The results pane shows the output:

```

International Number
Statement processed.

0.00 seconds

```

At the bottom right, it says "Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved."

## ● Subqueries

- 1. Write a PL/SQL block that uses a subquery to find the ho\_name of the homeowner who has the highest total\_amount in the Transaction table.**

DECLARE

```

v_ho_name VARCHAR2(100);

BEGIN
    SELECT ho_name INTO v_ho_name
    FROM (SELECT ho_name
          FROM Homeowner h
          JOIN Transaction t ON h.ho_id = t.ho_id
          GROUP BY ho_name
          ORDER BY SUM(amount) DESC)
          WHERE ROWNUM = 1;

```

```
        ORDER BY t.total_amount DESC)  
WHERE ROWNUM = 1;
```

```
    DBMS_OUTPUT.PUT_LINE('Homeowner with highest transaction amount: ' ||  
v_ho_name);  
END;  
/  
/
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```
DECLARE  
v_ho_name VARCHAR2(100);  
BEGIN  
SELECT ho_name INTO v_ho_name  
FROM (SELECT ho_name  
      FROM Homeowner h  
      JOIN Transaction t ON h.ho_id = t.ho_id  
      ORDER BY t.total_amount DESC)  
WHERE ROWNUM = 1;  
DBMS_OUTPUT.PUT_LINE('Homeowner with highest transaction amount: ' || v_ho_name);  
END;  
/
```

The results pane displays the output:

```
Homeowner with highest transaction amount:  
Statement processed.  
0.00 seconds
```

At the bottom, the status bar shows "Application Express 2.1 0.00.39" and "Copyright © 1999, 2005, Oracle. All rights reserved."

2. Write a PL/SQL block that uses a subquery to find the he\_name of the helper who has completed the most number of jobs in the Job table.

```
DECLARE  
v_he_name VARCHAR2(100);  
BEGIN  
SELECT he_name INTO v_he_name  
FROM (SELECT he_name  
      FROM Helper h  
      JOIN Job j ON h.he_id = j.he_id  
      GROUP BY h.he_name  
      ORDER BY COUNT(j.job_id) DESC)  
WHERE ROWNUM = 1;
```

```
        ORDER BY COUNT(*) DESC)
```

```
WHERE ROWNUM = 1;
```

```
DBMS_OUTPUT.PUT_LINE('Helper with most completed jobs: ' || v_he_name);
```

```
END;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code entered is:

```
DECLARE
  v_he_name VARCHAR2(100);
BEGIN
  SELECT he_name INTO v_he_name
  FROM (SELECT he_name
         FROM Helper h
         JOIN Job j ON h.he_id = j.he_id
        GROUP BY h.he_name
        ORDER BY COUNT(*) DESC)
  WHERE ROWNUM = 1;
  DBMS_OUTPUT.PUT_LINE('Helper with most completed jobs: ' || v_he_name);
END;
/
```

The results pane shows the output:

```
Helper with most completed jobs:  
Statement processed.  
0.01 seconds
```

At the bottom, it says "Language: en-us" and "Application Express 2.1.0.00.39 Copyright © 1995, 2006, Oracle. All rights reserved."

## ● Joining

1. Write a PL/SQL block to retrieve the names of homeowners and the corresponding helper names for all jobs where the job status is 'Open'.

```
DECLARE
```

```
  v_ho_name VARCHAR2(100);
```

```
  v_he_name VARCHAR2(100);
```

```
BEGIN
```

```
  FOR job IN (SELECT h.ho_name, he.he_name
```

```

        FROM Homeowners h
        JOIN Job j ON h.ho_id = j.ho_id
        JOIN Helper he ON j.he_id = he.he_id
        WHERE j.status = 'Open') LOOP
        v_ho_name := job.ho_name;
        v_he_name := job.he_name;
        DBMS_OUTPUT.PUT_LINE('Homeowner: ' || v_ho_name || ', Helper: ' || v_he_name);
    END LOOP;
END;

```

```

DECLARE
    v_ho_name VARCHAR2(100);
    v_he_name VARCHAR2(100);
BEGIN
    FOR job IN (SELECT h.ho_name, he.he_name
                FROM Homeowner h
                JOIN Job j ON h.ho_id = j.ho_id
                JOIN Helper he ON j.he_id = he.he_id
                WHERE j.status = 'Open') LOOP
        v_ho_name := job.ho_name;
        v_he_name := job.he_name;
        DBMS_OUTPUT.PUT_LINE('Homeowner: ' || v_ho_name || ', Helper: ' || v_he_name);
    END LOOP;
END;
/

```

Results Explain Describe Saved SQL History

Homeowner: , Helper:  
Homeowner: , Helper:  
Homeowner: , Helper:

Statement processed.

0.00 seconds

Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

**2. Write a PL/SQL block to retrieve the details of all transactions along with the corresponding homeowner names, where the transaction amount is greater than 200.**

```

DECLARE
    v_ho_name VARCHAR2(100);
    v_transaction_id NUMBER;
    v_transaction_amount NUMBER;

```

```

BEGIN
FOR t IN (SELECT h.ho_name, t.transaction_id, t.total_amount
          FROM Homeowner h
         JOIN Transaction t ON h.ho_id = t.ho_id
        WHERE t.total_amount > 200) LOOP
    v_ho_name := t.ho_name;
    v_transaction_id := t.transaction_id;
    v_transaction_amount := t.total_amount;
    DBMS_OUTPUT.PUT_LINE('Transaction ID: ' || v_transaction_id ||
                         ', Homeowner: ' || v_ho_name ||
                         ', Amount: ' || v_transaction_amount);
END LOOP;
END;
/

```

The screenshot shows the Oracle Database Express Edition interface. The title bar reads "127.0.0.1:8080/apex/f?p=4500:1003:339858763061637::NO::". The main area displays the following PL/SQL code:

```

ORACLE Database Express Edition
User: SCOTT
Home > SQL > SQL Commands

FOR t IN (SELECT h.ho_name, t.transaction_id, t.total_amount
          FROM Homeowner h
         JOIN Transaction t ON h.ho_id = t.ho_id
        WHERE t.total_amount > 200) LOOP
    v_ho_name := t.ho_name;
    v_transaction_id := t.transaction_id;
    v_transaction_amount := t.total_amount;
    DBMS_OUTPUT.PUT_LINE('Transaction ID: ' || v_transaction_id ||
                         ', Homeowner: ' || v_ho_name ||
                         ', Amount: ' || v_transaction_amount);
END LOOP;
END;
/

```

Below the code, the results pane shows the output of the DBMS\_OUTPUT.PUT\_LINE statements:

```

Transaction ID: 2, Homeowner: , Amount: 220.5
Transaction ID: 4, Homeowner: , Amount: 330.25
Statement processed.

0.00 seconds

```

The bottom right corner of the window displays the Application Express version: "Application Express 2.1.0.00.39".

## **Advanced Query Writing (PL/SQL):**

### **● Stored Function:**

1. Write a PL/SQL stored function that takes a ho\_id as an input parameter and returns the total amount of all transactions for that homeowner.

```
CREATE OR REPLACE FUNCTION get_total_transaction_amount(p_ho_id IN
NUMBER)
RETURN NUMBER
IS
v_total_amount NUMBER := 0;
BEGIN
SELECT SUM(total_amount)
INTO v_total_amount
FROM Transaction
WHERE ho_id = p_ho_id;

RETURN v_total_amount;
END;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. A user named SCOTT is logged in. In the central workspace, a SQL command is being entered:

```

CREATE OR REPLACE FUNCTION get_job_status(p_job_id IN NUMBER)
RETURN VARCHAR2
IS
    v_status VARCHAR2(20);
BEGIN
    SELECT status
    INTO v_status
    FROM Job
    WHERE job_id = p_job_id;
    RETURN v_status;
END;
/

```

Below the code, the message "Function created." is displayed, followed by "0.00 seconds". At the bottom right, the Application Express version is noted as "Application Express 2.1.0.0.39".

## 2. Write a PL/SQL stored function that takes a **job\_id** as an input parameter and returns the status of the job.

```
CREATE OR REPLACE FUNCTION get_job_status(p_job_id IN NUMBER)
```

```
    RETURN VARCHAR2
```

```
IS
```

```
    v_status VARCHAR2(20);
```

```
BEGIN
```

```
    SELECT status
```

```
    INTO v_status
```

```
    FROM Job
```

```
    WHERE job_id = p_job_id;
```

```
    RETURN v_status;
```

```
END;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. A user named SCOTT is connected. In the SQL editor, a function named `get_job_status` is being created. The code is as follows:

```

CREATE OR REPLACE FUNCTION get_job_status(p_job_id IN NUMBER)
  RETURN VARCHAR2
IS
  v_status VARCHAR2(20);
BEGIN
  SELECT status
  INTO v_status
  FROM Job
  WHERE job_id = p_job_id;
  RETURN v_status;
END;
/

```

The results pane shows the message "Function created." and a duration of "0.00 seconds". The bottom status bar indicates "Language: en-us" and "Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved." The system tray at the bottom right shows the date and time as "12/14/2024 7:18 PM".

## ● Stored Procedure:

- 1. Write a PL/SQL stored procedure that takes a ho\_id as an input parameter and updates the homeowner's phone number in the Homeowner table. The procedure should also print a message indicating whether the phone number was updated successfully or not.**

```
CREATE OR REPLACE PROCEDURE update_homeowner_phone(p_ho_id IN NUMBER, p_new_phone IN VARCHAR2)
```

```
IS
```

```
BEGIN
```

```
    UPDATE Homeowner
```

```
        SET ho_phone = p_new_phone
```

```
        WHERE ho_id = p_ho_id;
```

```
    IF SQL%ROWCOUNT > 0 THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Phone number updated successfully for
Homeowner ID: ' || p_ho_id);
```

```

ELSE
    DBMS_OUTPUT.PUT_LINE('No record found for Homeowner ID: ' || p_ho_id);
END IF;
END;

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```

CREATE OR REPLACE PROCEDURE update_homeowner_phone(p_ho_id IN NUMBER, p_new_phone IN VARCHAR2)
IS
BEGIN
    UPDATE Homeowner
    SET ho_phone = p_new_phone
    WHERE ho_id = p_ho_id;

    IF SQLROWCOUNT > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Phone number updated successfully for Homeowner ID: ' || p_ho_id);
    ELSE
        DBMS_OUTPUT.PUT_LINE('No record found for Homeowner ID: ' || p_ho_id);
    END IF;
END;
/

```

The results pane shows the message "Procedure created." and a execution time of "0.01 seconds". The status bar at the bottom right indicates "Application Express 2.1.0.0939 Copyright © 1999, 2006, Oracle. All rights reserved".

**2. Write a PL/SQL stored procedure that takes a job\_id as an input parameter and deletes the corresponding job from the Job table. The procedure should raise an exception if no job with the given job\_id exists in the table.**

```
CREATE OR REPLACE PROCEDURE delete_job(p_job_id IN NUMBER)
```

```
IS
```

```
    v_count NUMBER;
```

```
BEGIN
```

```
    SELECT COUNT(*) INTO v_count
```

```
    FROM Job
```

```
    WHERE job_id = p_job_id;
```

```
    IF v_count = 0 THEN
```

```

    RAISE_APPLICATION_ERROR(-20001, 'Job with ID ' || p_job_id || ' does not
exist.');

    ELSE

        DELETE FROM Job

        WHERE job_id = p_job_id;

        DBMS_OUTPUT.PUT_LINE('Job with ID ' || p_job_id || ' deleted successfully.);

        END IF;

    END;

/

```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a PL/SQL procedure named 'delete\_job' is being created. The code checks if a job exists by counting rows in the 'Job' table where 'job\_id' matches the input parameter 'p\_job\_id'. If no rows are found, it raises an application error. Otherwise, it deletes the row and outputs a success message. The procedure ends with a slash ('/'). Below the code, the results show 'Procedure created.' and a execution time of '0.00 seconds'. The bottom status bar indicates the application version is 'Application Express 2.1.0.00.39' and the copyright year is 'Copyright © 1999, 2006, Oracle. All rights reserved.'

```

CREATE OR REPLACE PROCEDURE delete_job(p_job_id IN NUMBER)
IS
    v_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_count
    FROM Job
    WHERE job_id = p_job_id;

    IF v_count = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Job with ID ' || p_job_id || ' does not exist.');
    ELSE
        DELETE FROM Job
        WHERE job_id = p_job_id;
        DBMS_OUTPUT.PUT_LINE('Job with ID ' || p_job_id || ' deleted successfully.');
    END IF;
END;
/

```

## ● Time-based Record

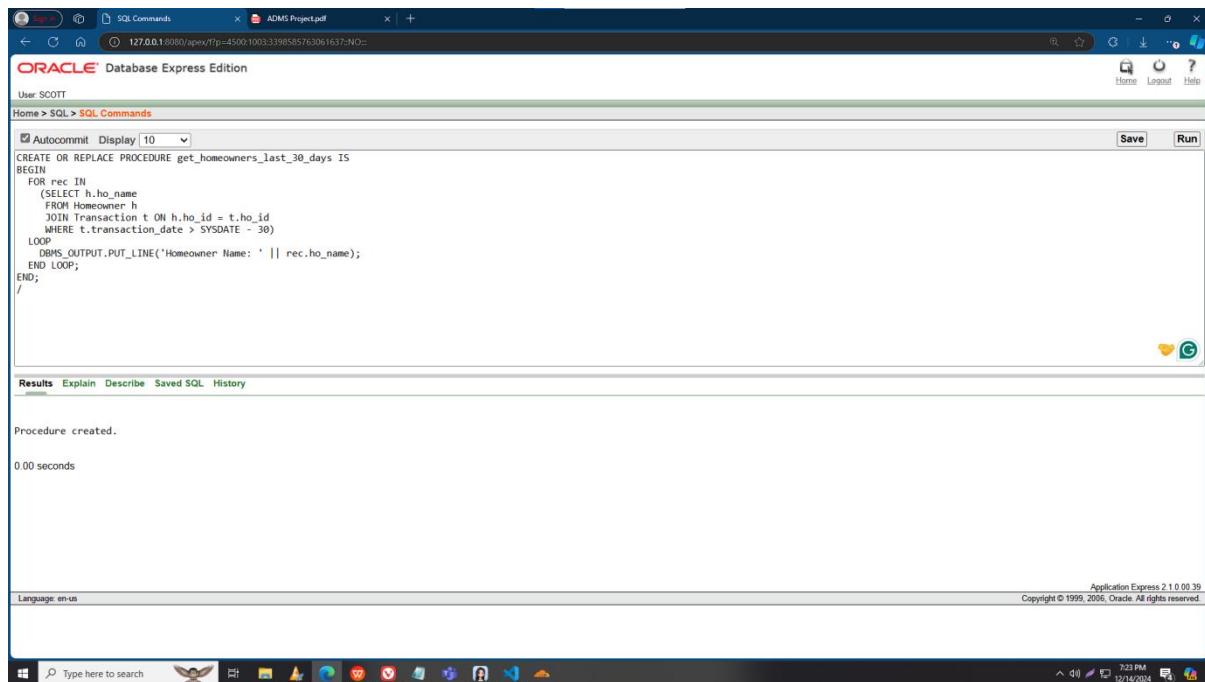
1. Write a PL/SQL stored procedure to retrieve the names of homeowners who have made transactions in the last 30 days. The procedure should accept no input parameters and should display the names of these homeowners.

```
CREATE OR REPLACE PROCEDURE get_homeowners_last_30_days IS
```

```
BEGIN
```

```
FOR rec IN
```

```
(SELECT h.ho_name
  FROM Homeowner h
  JOIN Transaction t ON h.ho_id = t.ho_id
 WHERE t.transaction_date > SYSDATE - 30)
LOOP
  DBMS_OUTPUT.PUT_LINE('Homeowner Name: ' || rec.ho_name);
END LOOP;
END;
```



2. Write a PL/SQL stored procedure to display the names of helpers who have completed at least one job in the last 7 days. The procedure should accept no parameters and should display the names of these helpers.

```
CREATE OR REPLACE PROCEDURE get_helpers_last_7_days IS  
BEGIN  
FOR rec IN  
(SELECT h.he_name  
FROM Helper h
```

```

        JOIN Job j ON h.he_id = j.he_id
        WHERE j.postdate > SYSDATE - 7)
    LOOP
        DBMS_OUTPUT.PUT_LINE('Helper Name: ' || rec.he_name);
    END LOOP;
END;

```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a PL/SQL procedure is being created:

```

CREATE OR REPLACE PROCEDURE get_helpers_last_7_days IS
BEGIN
FOR rec IN
  (SELECT h.he_name
   FROM Helper h
   JOIN Job j ON h.he_id = j.he_id
   WHERE j.postdate > SYSDATE - 7)
LOOP
DBMS_OUTPUT.PUT_LINE('Helper Name: ' || rec.he_name);
END LOOP;
END;
/

```

After executing the code, the message "Procedure created." is displayed. The system status bar at the bottom right shows "Application Express 2.1.0.09.39" and "Copyright © 1999, 2005, Oracle. All rights reserved."

## ● Explicit Cursor

1. Write a PL/SQL block that uses an explicit cursor to retrieve the total transaction amount for each homeowner from the Transaction table. The block should output the ho\_name and the corresponding total amount.

```

DECLARE
    CURSOR c_transactions IS
        SELECT h.ho_name, SUM(t.total_amount) AS total_amount
        FROM Homeowner h

```

```

JOIN Transaction t ON h.ho_id = t.ho_id
GROUP BY h.ho_name;

v_ho_name Homeowner.ho_name%TYPE;
v_total_amount Transaction.total_amount%TYPE;

BEGIN
OPEN c_transactions;
LOOP
FETCH c_transactions INTO v_ho_name, v_total_amount;
EXIT WHEN c_transactions%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('Homeowner: ' || v_ho_name || ' | Total Transaction
Amount: ' || v_total_amount);
END LOOP;
CLOSE c_transactions;
END;

```

```

SELECT h.ho_name, SUM(t.total_amount) AS total_amount
FROM Homeowner h
JOIN Transaction t ON h.ho_id = t.ho_id
GROUP BY h.ho_name;
v_ho_name Homeowner.ho_name%TYPE;
v_total_amount Transaction.total_amount%TYPE;
BEGIN
OPEN c_transactions;
LOOP
FETCH c_transactions INTO v_ho_name, v_total_amount;
EXIT WHEN c_transactions%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('Homeowner: ' || v_ho_name || ' | Total Transaction Amount: ' || v_total_amount);
END LOOP;
CLOSE c_transactions;
END;
/

```

Results Explain Describe Saved SQL History

Homeowner: | Total Transaction Amount: 1007.1  
Statement processed.  
0.00 seconds

Language: en-us Application Express 2.1.0.0.39  
Copyright © 1999, 2005, Oracle. All rights reserved.

2. Write a PL/SQL block that uses an explicit cursor to retrieve and display the names and ratings of helpers from the Helper table who have a rating greater than 4.5.

```
DECLARE
```

```
    CURSOR c_helpers IS
        SELECT he_name, he_rating
        FROM Helper
        WHERE he_rating > 4.5;
```

```
        v_he_name Helper.he_name%TYPE;
```

```
        v_he_rating Helper.he_rating%TYPE;
```

```
BEGIN
```

```
    OPEN c_helpers;
```

```
    LOOP
```

```
        FETCH c_helpers INTO v_he_name, v_he_rating;
```

```
        EXIT WHEN c_helpers%NOTFOUND;
```

```
        DBMS_OUTPUT.PUT_LINE('Helper: ' || v_he_name || ' | Rating: ' ||
v_he_rating);
```

```
    END LOOP;
```

```
    CLOSE c_helpers;
```

```
END;
```

```
/
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. A PL/SQL block is being run:

```

CURSOR c_helpers IS
SELECT he_name, he_rating
FROM Helper
WHERE he_rating > 4.5;
BEGIN
OPEN c_helpers;
LOOP
  FETCH c_helpers INTO v_he_name, v_he_rating;
  EXIT WHEN c_helpers%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE('Helper: ' || v_he_name || ' | Rating: ' || v_he_rating);
END LOOP;
CLOSE c_helpers;
END;
/

```

The results of the query are displayed below:

```

Helper: James Carter | Rating: 4.8
Helper: Maria Lopez | Rating: 4.7
Helper: Robert Williams | Rating: 4.9
Helper: Sophie Taylor | Rating: 4.6
Statement processed.
0.00 seconds

```

At the bottom, it says "Language: en-us" and "Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved." The system tray at the bottom right shows the date and time as 12/14/2024 7:29 PM.

## ● Cursor-Based Record

1. Write a PL/SQL block that uses a cursor-based record to retrieve all the homeowners (ho\_name) who have a pending payment status from the Payment table. The block should display the ho\_name and the payment details (i.e., p\_amount, p\_method, and Pay\_status).

DECLARE

```

CURSOR c_pending_payments IS
  SELECT h.ho_name, p.p_amount, p.p_method, p.Pay_status
    FROM Homeowner h
   JOIN Payment p ON h.ho_id = p.ho_id
 WHERE p.Pay_status = 'Pending';

```

TYPE PaymentRec IS RECORD (

```

  ho_name Homeowner.ho_name%TYPE,
  p_amount Payment.p_amount%TYPE,

```

```

    p_method Payment.p_method%TYPE,
    Pay_status Payment.Pay_status%TYPE
);

v_payment PaymentRec;

BEGIN
OPEN c_pending_payments;
LOOP
  FETCH c_pending_payments INTO v_payment;
  EXIT WHEN c_pending_payments%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE('Homeowner: ' || v_payment.ho_name ||
    ' | Amount: ' || v_payment.p_amount ||
    ' | Method: ' || v_payment.p_method ||
    ' | Status: ' || v_payment.Pay_status);
END LOOP;
CLOSE c_pending_payments;
END;

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL command window displays the following PL/SQL block:

```

    p_method Payment.p_method%TYPE,
    Pay_status Payment.Pay_status%TYPE
);

v_payment PaymentRec;

BEGIN
OPEN c_pending_payments;
LOOP
  FETCH c_pending_payments INTO v_payment;
  EXIT WHEN c_pending_payments%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE('Homeowner: ' || v_payment.ho_name ||
    ' | Amount: ' || v_payment.p_amount ||
    ' | Method: ' || v_payment.p_method ||
    ' | Status: ' || v_payment.Pay_status);
END LOOP;
CLOSE c_pending_payments;
END;
/

```

The results pane shows the output of the DBMS\_OUTPUT.PUT\_LINE statements:

```

Homeowner: | Amount: 200 | Method: Cash | Status: Pending
Statement processed.

0.00 seconds

```

At the bottom of the interface, the footer includes the Application Express version (2.1.0.0939), copyright information (Copyright © 1999, 2006, Oracle. All rights reserved.), and the system date (12/14/2014).

2. Write a PL/SQL block that uses a cursor-based record to retrieve the names of all helpers (he\_name) and the number of jobs they have completed from the Job table. Display the helper's name along with their job count.

**DECLARE**

```
CURSOR c_helper_jobs IS
  SELECT h.he_name, COUNT(j.job_id) AS job_count
  FROM Helper h
  JOIN Job j ON h.he_id = j.he_id
  WHERE j.status = 'Closed'
  GROUP BY h.he_name;
```

```
TYPE HelperJobRec IS RECORD (
```

```
  he_name Helper.he_name%TYPE,
  job_count NUMBER
);
```

```
v_helper_job HelperJobRec;
```

**BEGIN**

```
  OPEN c_helper_jobs;
```

**LOOP**

```
  FETCH c_helper_jobs INTO v_helper_job;
  EXIT WHEN c_helper_jobs%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE('Helper: ' || v_helper_job.he_name ||
    ' | Completed Jobs: ' || v_helper_job.job_count);
```

**END LOOP;**

```
  CLOSE c_helper_jobs;
```

**END;**

/

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code in the editor window is:

```

TYPE HelperJobRec IS RECORD (
    he_name Helper.he_name%TYPE,
    job_count NUMBER
);

v_helper_job HelperJobRec;
BEGIN
OPEN c_helper_jobs;
LOOP
  FETCH c_helper_jobs INTO v_helper_job;
  EXIT WHEN c_helper_jobs%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE('Helper: ' || v_helper_job.he_name || '| Completed Jobs: ' || v_helper_job.job_count);
END LOOP;
CLOSE c_helper_jobs;
END;
/

```

The results window shows the output of the script:

```

Helper: | Completed Jobs: 2
Statement processed.

0.02 seconds

```

At the bottom right of the interface, it says "Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved."

## ● Row-Level Trigger

**1. Create a row-level trigger on the Payment table that will prevent a payment from being inserted if the payment amount (p\_amount) is less than 50. The trigger should raise an error with a message: "Payment amount must be at least 50."**

CREATE OR REPLACE TRIGGER trg\_prevent\_small\_payment

BEFORE INSERT ON Payment

FOR EACH ROW

BEGIN

IF :NEW.p\_amount < 50 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Payment amount must be at least 50');

END IF;

END;

The screenshot shows the Oracle Database Express Edition SQL Commands interface. A trigger named 'trg\_prevent\_small\_payment' is being created. The code is as follows:

```
CREATE OR REPLACE TRIGGER trg_prevent_small_payment
BEFORE INSERT ON Payment
FOR EACH ROW
BEGIN
  IF :NEW.p_amount < 50 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Payment amount must be at least 50');
  END IF;
END;
/
```

The results pane shows the message "Trigger created." and a duration of "0.01 seconds". The bottom status bar indicates "Language: en-us" and "Application Express 2.1 0 00 39 Copyright © 1999, 2006, Oracle. All rights reserved."

2. Create a row-level trigger on the Payment table that will automatically update the status of the associated Job to 'Paid' when a payment is inserted into the Payment table. The trigger should fire after an insert operation on the Payment table.

CREATE OR REPLACE TRIGGER trg\_update\_job\_status

AFTER INSERT ON Payment

FOR EACH ROW

BEGIN

UPDATE Job

SET status = 'Paid'

WHERE job\_id = :NEW.job\_id;

END;

/

The screenshot shows the Oracle Database Express Edition SQL Commands interface. A user named SCOTT is connected. In the SQL editor, a trigger named 'trg\_update\_job\_status' is being created:

```

CREATE OR REPLACE TRIGGER trg_update_job_status
AFTER INSERT ON Payment
FOR EACH ROW
BEGIN
  UPDATE Job
  SET status = 'Paid'
  WHERE job_id = :NEW.job_id;
END;
/

```

The results pane shows the message "Trigger created." and a duration of "0.00 seconds". The bottom status bar indicates "Application Express 2.1 0.00 39 Copyright © 1999, 2006, Oracle. All rights reserved." The taskbar at the bottom shows various application icons.

## ● Statement level Trigger:

**1. Create a statement-level trigger that tracks every insert operation on the Job table. The trigger should output a message to the console that states: "Number of records inserted into Job table: X", where X is the number of rows inserted. You do not need to create a separate table to log this information.**

```
CREATE OR REPLACE TRIGGER trg_log_job_inserts
```

```
AFTER INSERT ON Job
```

```
DECLARE
```

```
    v_count NUMBER;
```

```
BEGIN
```

```
    SELECT COUNT(*) INTO v_count FROM Job;
```

```

DBMS_OUTPUT.PUT_LINE('Number of records inserted into
Job table: ' || v_count);

END;

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code entered is:

```

CREATE OR REPLACE TRIGGER trg_log_job_inserts
AFTER INSERT ON Job
DECLARE
  v_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_count FROM Job;
  DBMS_OUTPUT.PUT_LINE('Number of records inserted into Job table: ' || v_count);
END;
/

```

The results pane shows the output:

```

Trigger created.

0.00 seconds

```

At the bottom, it says "Language: en-us" and "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved." The system tray at the bottom right shows the date as 12/14/2024 and the time as 7:41 PM.

**2. Create a statement-level trigger that tracks every update operation on the Payment table. The trigger should output a message to the console stating: "Update operation performed on Payment table." The message should also include the number of records updated in the Payment table.**

```

CREATE OR REPLACE TRIGGER trg_log_payment_updates
AFTER UPDATE ON Payment
DECLARE
  v_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_count FROM Payment WHERE ROWNUM <= 1;

```

```

DBMS_OUTPUT.PUT_LINE('Update operation performed on Payment table.');

DBMS_OUTPUT.PUT_LINE('Number of records updated in Payment table: ' ||
v_count);

END;

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL editor contains the following PL/SQL code:

```

CREATE OR REPLACE TRIGGER trg_log_payment_updates
AFTER UPDATE ON Payment
DECLARE
    v_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_count FROM Payment WHERE ROWNUM <= 1;
    DBMS_OUTPUT.PUT_LINE('Update operation performed on Payment table.');
    DBMS_OUTPUT.PUT_LINE('Number of records updated in Payment table: ' || v_count);
END;
/

```

The results pane shows the message "Trigger created." and "0.00 seconds". The bottom status bar indicates "Language: en-us" and "Application Express 2.1.0.0.39 Copyright © 1999, 2005, Oracle. All rights reserved".

## ● Package

### 1. Create a package to retrieve homeowner details.

```

CREATE OR REPLACE PACKAGE homeowner_pkg IS

PROCEDURE get_homeowner_details(p_ho_id IN NUMBER);

FUNCTION get_homeowner_email(p_ho_id IN NUMBER)
RETURN VARCHAR2;

END homeowner_pkg;

```

```
CREATE OR REPLACE PACKAGE BODY homeowner_pkg IS

    PROCEDURE get_homeowner_details(p_ho_id IN NUMBER)
    IS
        v_ho_name      VARCHAR2(100);
        v_ho_address  VARCHAR2(255);
        v_ho_email     VARCHAR2(100);
        v_ho_phone     VARCHAR2(20);

    BEGIN
        SELECT ho_name, ho_address, ho_email, ho_phone
        INTO v_ho_name, v_ho_address, v_ho_email, v_ho_phone
        FROM Homeowner
        WHERE ho_id = p_ho_id;

        DBMS_OUTPUT.PUT_LINE('Homeowner Details:');
        DBMS_OUTPUT.PUT_LINE('Name: ' || v_ho_name);
        DBMS_OUTPUT.PUT_LINE('Address: ' || v_ho_address);
        DBMS_OUTPUT.PUT_LINE('Email: ' || v_ho_email);
        DBMS_OUTPUT.PUT_LINE('Phone: ' || v_ho_phone);

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('No homeowner found
with ID ' || p_ho_id);
```

```
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred.');
END get_homeowner_details;

FUNCTION get_homeowner_email(p_ho_id IN NUMBER)
RETURN VARCHAR2 IS
    v_ho_email VARCHAR2(100);
BEGIN
    SELECT ho_email
    INTO v_ho_email
    FROM Homeowner
    WHERE ho_id = p_ho_id;
    RETURN v_ho_email;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
    WHEN OTHERS THEN
        RETURN NULL;
END get_homeowner_email;
END homeowner_pkg;
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. A package named 'homeowner\_pkg' is being created. The code defines a function 'get\_homeowner\_email' that takes a parameter 'p\_ho\_id' and returns a VARCHAR2 value. It uses a cursor to select the email from the 'Homeowner' table where the ho\_id matches the input parameter. If no data is found, it returns NULL. The package body is successfully created in 0.00 seconds.

```

FUNCTION get_homeowner_email(p_ho_id IN NUMBER) RETURN VARCHAR2 IS
  v_ho_email VARCHAR2(100);
BEGIN
  SELECT ho_email
  INTO v_ho_email
  FROM Homeowner
  WHERE ho_id = p_ho_id;
  RETURN v_ho_email;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN NULL;
  WHEN OTHERS THEN
    RETURN NULL;
END get_homeowner_email;
END homeowner_pkg;
/

```

Package Body created.  
0.00 seconds

Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## 2. Create a package to calculate helper performance.

```

CREATE OR REPLACE PACKAGE helper_performance_pkg IS
  PROCEDURE update_performance(p_he_id IN NUMBER, p_rating IN NUMBER);
  FUNCTION get_average_rating(p_he_id IN NUMBER) RETURN NUMBER;
END helper_performance_pkg;

```

```
CREATE OR REPLACE PACKAGE BODY helper_performance_pkg IS
```

```
  PROCEDURE update_performance(p_he_id IN NUMBER, p_rating IN NUMBER) IS
  BEGIN
```

```
    UPDATE Helper
```

```
    SET he_rating = p_rating
```

```
    WHERE he_id = p_he_id;
```

```
    COMMIT;
```

```
    DBMS_OUTPUT.PUT_LINE('Helper performance updated for ID ' || p_he_id);
```

```

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No helper found with ID ' || p_he_id);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred while updating
performance.');
END update_performance;

FUNCTION get_average_rating(p_he_id IN NUMBER) RETURN NUMBER IS
    v_avg_rating NUMBER;
BEGIN
    SELECT AVG(he_rating)
    INTO v_avg_rating
    FROM Helper
    WHERE he_id = p_he_id;

    RETURN v_avg_rating;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
    WHEN OTHERS THEN
        RETURN NULL;
END get_average_rating;
END helper_performance_pkg;

```

SQL Commands    ADMs Project.pdf

ORACLE Database Express Edition

User SCOTT

Home > SQL > SQL Commands

Autocommit Display 10

```
FUNCTION get_homeowner_email(p_ho_id IN NUMBER) RETURN VARCHAR2 IS
  v_ho_email VARCHAR2(100);
BEGIN
  SELECT ho_email
  INTO v_ho_email
  FROM Homeowner
  WHERE ho_id = p_ho_id;
  RETURN v_ho_email;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN NULL;
  WHEN OTHERS THEN
    RETURN NULL;
END get_homeowner_email;
/
```

Save Run

Results Explain Describe Saved SQL History

Package Body created.

0.00 seconds

Language: en-us Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

## **Conclusion**

Developed a comprehensive database schema for an helping hand agency.

Created sample data and executed SQL queries for data retrieval. Explored relational algebra for database query formulation.

Our planned future work:

- Improve data analysis capabilities with complex SQL queries.
- Develop a user-friendly interface for easier database interaction.
- Enhance security measures for data protection.
- Optimize database performance for scalability.