



AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)

Department of the Faculty of Science and Technology

Summer 2024-2025

**Final Term Project Documentations on
Laundry Service Management System**

**Supervised By
JUENA AHMED NOSHIN**

Name	ID	Contribution
Tanvir Arabi	22-47857-2	Code implementation, Query implementation, and Data insertion (35%)
Bushra Kaiser	22-47679-2	Code implementation, Normalization, Query implementation, and Relational Algebra (35%)
MD. Anas Khan	20-43091-1	Code implementation, Query implementation, and Data insertion (30%)

ADVANCED DATABASE MANAGEMENT SYSTEM

Section: B

Group No: 04

Contents:

Content Name	Page No
Normalization	2-5
Table creation & data insertion	5-16
Missing PL/SQL query	17-42
Exception Handling	43-73
Relational Algebra	73-74
User interface & Code	74-78

Project Updates

1. Normalization:

1) SERVE

UNF:

1. serve (c_id, c_name, c_email, c_phone, c_address, l_id, l_phone, l_address)

1NF:

c_phone and c_email are multi-valued attributes.

1. c_id, c_name, c_email, c_phone, c_address, l_id, l_phone, l_address

2NF:

1. c_id, c_name, c_email, c_phone, c_address

2. l_id, l_phone, l_address

3NF:

There is no transitive dependency. Relation already in 3NF .

1. c_id, c_name, c_email, c_phone, c_address

2. l_id, l_phone, l_address

Table Creation:

1. c_id, c_name, c_email, c_phone, c_address

2. l_id, l_phone, l_address, c_id .

2) HAVE

UNF:

1. Have (l_id, l_phone, l_address, e_id, e_name, e_dob, e_phone, e_address)

1NF:

There is no multi valued attribute. Relation already in 1NF.

1. l_id, l_phone, l_address, e_id, e_name, e_dob, e_phone, e_address

2NF:

1. e_id, e_name, e_dob, e_phone, e_address

2. l_id, l_phone, l_address

3NF:

There is no transitive dependency. Relation already in 3NF.

1. e_id, e_name, e_phone, e_address

2. l_id, l_phone, l_address

Table Creation :

1. e_id, e_name, e_phone, e_address

2. l_id, l_phone, l_address, e_id

3) PROVIDED

UNF:

Provided (**c_id**, c_name, c_email, c_phone, c_address, m_id, m_name, m_phone)

1NF:

c_phone and c_email are multi-valued attributes.

1. c_id, c_name, c_email, c_phone, c_address, m_id, m_name, m_phone

2NF:

1. **c_id**, c_name, c_email, c_phone, c_address,

2. **m_id**, m_name, m_phone

3NF:

There is no transitive dependency. Relation already in 3NF.

1. **c_id**, c_name, c_email, c_phone, c_address

2. **m_id**, m_name, m_phone

Table Creation:

1. **c_id**, c_name, c_email, c_phone, c_address

2. **m_id**, m_name, m_phone

3. **c_id**, **m_id**

4) PAY**UNF:**

Pay (**c_id**, c_name, c_email, c_phone, c_address, **u_id**, u_paytype, u_due)

1NF:

c_phone and c_email are multi-valued attributes.

1. **c_id**, c_name, c_email, c_phone, c_address, **u_id**, u_paytype, u_due.

2NF:

1. **c_id**, c_name, c_email, c_phone, c_address

2. **u_id**, u_paytype, u_due.

3NF:

There is no transitive dependency. Relation already in 3NF.

1. **c_id**, c_name, c_email, c_phone, c_address

2. **u_id**, u_paytype, u_due.

Table Creation:

1. **c_id**, c_name, c_email, c_phone, c_address

2. **u_id**, u_paytype, u_due.

3. **c_id**, **u_id**

5) OWNS**UNF:**

owns (**o_id**, o_name, o_email, o_phone, o_address, **l_id**, l_phone, l_address)

1NF:

There is no multi-valued attribute. Relation already in 1NF.

1. **o_id**, o_name, o_email, o_phone, o_address, **l_id**, l_phone, l_address

2NF:

1. **o_id**, o_name, o_email, o_phone, o_address

2. **l_id**, l_phone, l_address

3NF:

There is no transitive dependency. Relation already in 3NF.

1. **o_id**, o_name, o_email, o_phone, o_address

2. **l_id**, l_phone, l_address

Table Creation :

1. **o_id**, o_name, o_email, o_phone, o_address

2. **l_id**, l_phone, l_address

3. **o_id**, **l_id**

6) OFFER

UNF:

offer (**l_id**, l_phone, l_address, **h_id**, h_pdate, h_ddate, h_address)

1NF:

There is no multi-valued attribute. Relation already in 1NF.

1. **l_id**, l_phone, l_address, **h_id**, h_pdate, h_ddate, h_address

2NF:

1. **h_id**, h_pdate, h_ddate, h_address

2. **l_id**, l_phone, l_address

3NF:

There is no transitive dependency. Relation already in 3NF.

1. **h_id**, h_pdate, h_ddate, h_address

2. **l_id**, l_phone, l_address

Table Creation:

1. **h_id**, h_pdate, h_ddate, h_address

2. **l_id**, l_phone, l_address ,**h_id**

Temporary table :

1. **c_id**, c_name, c_email, c_phone, c_address
2. **l_id**, l_phone, l_address, **c_id** .
3. **e_id**, e_name, e_phone, e_address
4. **l_id**, l_phone, l_address, **e_id**
5. **c_id**, c_name, c_email, c_phone, c_addressx
6. **m_id**, m_name, m_phone
7. **c_id**, **m_id**
8. **c_id**, c_name, c_email, c_phone, c_addressx

9. **u_id**, u_paytype, u_due.
10. **c_id**, **u_id**
11. **o_id**, o_name, o_email, o_phone, o_address
12. **l_id**, l_phone, l_address
13. **o_id**, **l_id**
14. **h_id**, h_pdate, h_ddate, h_address
15. **l_id**, l_phone, l_address ,**h_id**

Final tables

1. **c_id**, c_name, c_email, c_phone, c_address
2. **l_id**, l_phone, l_address, **c_id**, **e_id**, **h_id**
3. **e_id**, e_name, e_phone, e_address
4. **m_id**, m_name, m_phone
5. **c_id**, **m_id**
6. **u_id**, u_paytype, u_due.
7. **c_id**, **u_id**
8. **o_id**, o_name, o_email, o_phone, o_address
9. **o_id**, **l_id**
10. **h_id**, h_pdate, h_ddate, h_address

Table Name:

1. Customer
2. Laundry_Service
3. Employee
4. Membership_Card
5. Customer_Membership
6. Utilities_Bill
7. Customer_Utilsities
8. Owner
9. Owner_Laundry_Service
10. Home_Service

Table Creation Using SQL:

1.Customer:

```
CREATE TABLE Customer (
customer_id NUMBER(10) PRIMARY KEY,
first_name VARCHAR2(50) NOT NULL,
last_name VARCHAR2(50) NOT NULL,
phone_number VARCHAR2(15),
email VARCHAR2(100) UNIQUE,
address VARCHAR2(255));
```

```

CREATE TABLE Customer (
    customer_id NUMBER(10) PRIMARY KEY,
    first_name VARCHAR2(50) NOT NULL,
    last_name VARCHAR2(50) NOT NULL,
    phone_number VARCHAR2(15),
    email VARCHAR2(100) UNIQUE,
    address VARCHAR2(255)
);

```

Table created.

0.06 seconds



2. Laundry Service

```

CREATE TABLE Laundry_Service (
    service_id NUMBER(10) PRIMARY KEY,
    service_name VARCHAR2(100) NOT NULL,
    description VARCHAR2(500),
    price NUMBER(10, 2) NOT NULL
);

```

```

CREATE TABLE Laundry_Service (
    service_id NUMBER(10) PRIMARY KEY,
    service_name VARCHAR2(100) NOT NULL,
    description VARCHAR2(500),
    price NUMBER(10, 2) NOT NULL
);

```

Table created.

0.02 seconds



3. Employee

```

CREATE TABLE Employee (
    employee_id NUMBER (10) PRIMARY KEY,
    first_name VARCHAR2(50) NOT NULL,
    last_name VARCHAR2(50) NOT NULL,
    job_title VARCHAR2(100),

```

```
hire_date DATE,
salary NUMBER (10, 2)
);
```

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

```
CREATE TABLE Employee (
    employee_id NUMBER(10) PRIMARY KEY,
    first_name VARCHAR2(50) NOT NULL,
    last_name VARCHAR2(50) NOT NULL,
    job_title VARCHAR2(100),
    hire_date DATE,
    salary NUMBER(10, 2)
);
```

Below the code, the results show "Table created." and a execution time of "0.03 seconds".



4. Membership Card

```
CREATE TABLE Membership_Card (
    card_id NUMBER (10) PRIMARY KEY,
    card_type VARCHAR2(50) NOT NULL,
    discount_percentage NUMBER (5, 2),
    valid_until DATE);
```

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

```
CREATE TABLE Membership_Card (
    card_id NUMBER(10) PRIMARY KEY,
    card_type VARCHAR2(50) NOT NULL,
    discount_percentage NUMBER(5, 2),
    valid_until DATE
);
```

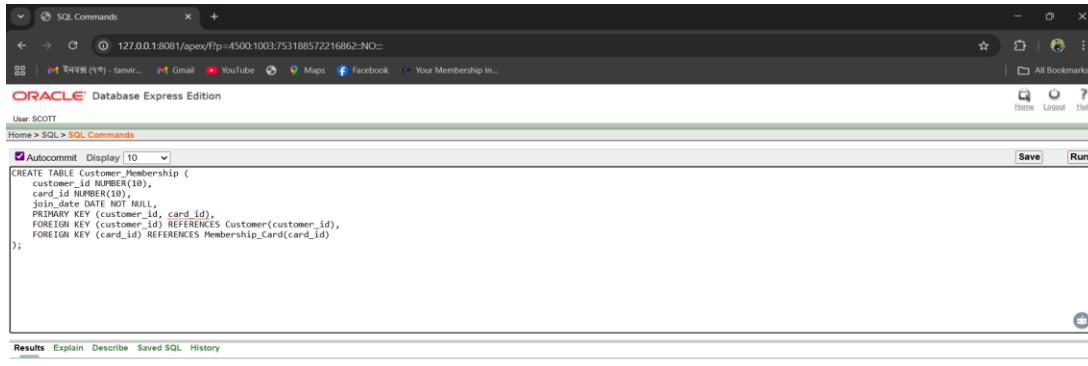
Below the code, the results show "Table created." and a execution time of "0.01 seconds".



5. Customer Membership

```
CREATE TABLE Customer_Membership (
    customer_id NUMBER(10),
    card_id NUMBER(10),
    join_date DATE NOT NULL,
```

PRIMARY KEY (customer_id, card_id),
 FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
 FOREIGN KEY (card_id) REFERENCES Membership_Card(card_id));



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

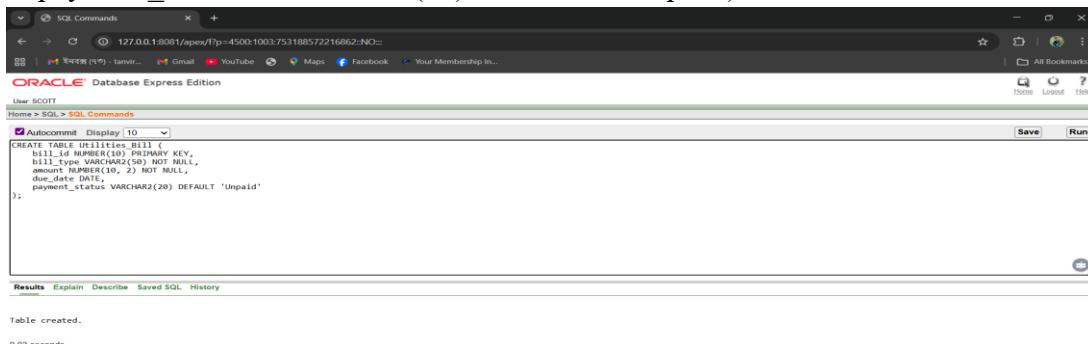
```
CREATE TABLE Customer_Membership (
  customer_id NUMBER(10),
  card_id NUMBER(10),
  join_date DATE DEFAULT NULL,
  PRIMARY KEY (customer_id, card_id),
  FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
  FOREIGN KEY (card_id) REFERENCES Membership_Card(card_id)
);
```

Below the code, the results show "Table created." and "0.03 seconds".



6. Utilities Bill

CREATE TABLE Utilities_Bill (bill_id NUMBER(10) PRIMARY KEY, bill_type VARCHAR2(50) NOT NULL, amount NUMBER(10, 2) NOT NULL, due_date DATE, payment_status VARCHAR2(20) DEFAULT 'Unpaid');



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

```
CREATE TABLE Utilities_Bill (
  bill_id NUMBER(10) PRIMARY KEY,
  bill_type VARCHAR2(50) NOT NULL,
  amount NUMBER(10, 2) NOT NULL,
  due_date DATE,
  payment_status VARCHAR2(20) DEFAULT 'Unpaid'
);
```

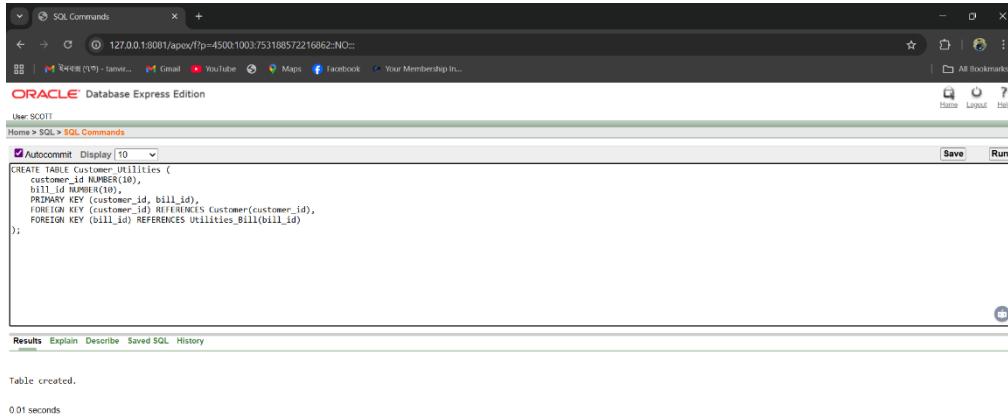
Below the code, the results show "Table created." and "0.03 seconds".



7.Customer Utilities

CREATE TABLE Customer_Utilsities (customer_id NUMBER(10), bill_id NUMBER(10), PRIMARY KEY (customer_id, bill_id),

FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
 FOREIGN KEY (bill_id) REFERENCES Utilities_Bill(bill_id));



```
SQL Commands
127.0.0.1:8081/apex/f?p=4500:1003:753188572216862::NO::
ORACLE Database Express Edition
User: SCOTT
Home > SQL > SQL Commands
Autocommit Display 10 ✓
CREATE TABLE Customer_Utilsities (
    customer_id NUMBER(10),
    bill_id NUMBER(10),
    PRIMARY KEY (customer_id, bill_id),
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id),
    FOREIGN KEY (bill_id) REFERENCES Utilities_Bill(bill_id)
);
Results Explain Describe Saved SQL History
```

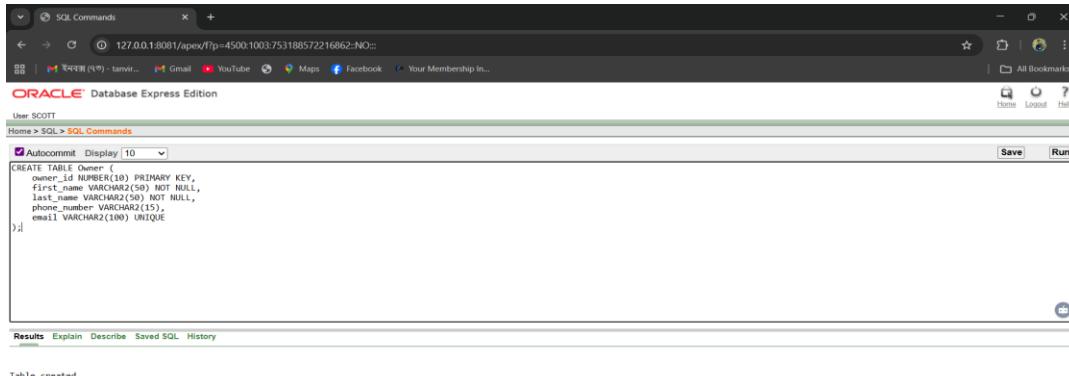
Table created.
0.01 seconds

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



8. Owner

CREATE TABLE Owner (
 owner_id NUMBER(10) PRIMARY KEY,
 first_name VARCHAR2(50) NOT NULL,
 last_name VARCHAR2(50) NOT NULL,
 phone_number VARCHAR2(15),
 email VARCHAR2(100) UNIQUE);



```
SQL Commands
127.0.0.1:8081/apex/f?p=4500:1003:753188572216862::NO::
ORACLE Database Express Edition
User SCOTT
Home > SQL > SQL Commands
Autocommit Display 10 ✓
CREATE TABLE Owner (
    owner_id NUMBER(10) PRIMARY KEY,
    first_name VARCHAR2(50) NOT NULL,
    last_name VARCHAR2(50) NOT NULL,
    phone_number VARCHAR2(15),
    email VARCHAR2(100) UNIQUE
);
Results Explain Describe Saved SQL History
```

Table created.
0.01 seconds

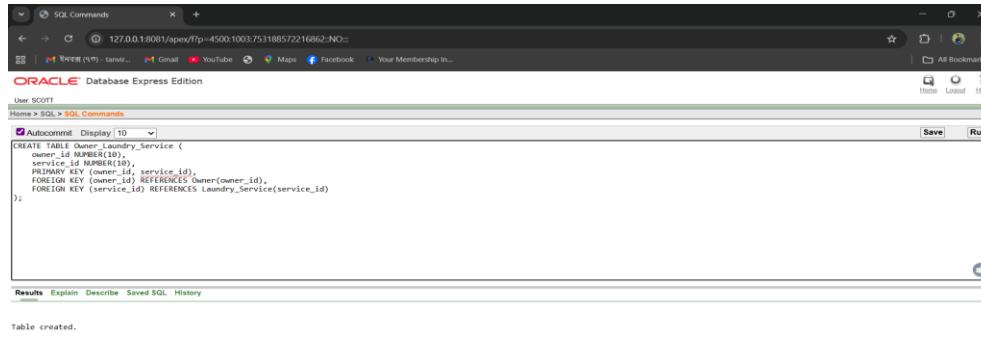
Language: en-gb Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved.



9. Owner Laundry Service:

CREATE TABLE Owner_Laundry_Service (
 owner_id NUMBER(10),
 service_id NUMBER(10),

PRIMARY KEY (owner_id, service_id),
 FOREIGN KEY (owner_id) REFERENCES Owner(owner_id),
 FOREIGN KEY (service_id) REFERENCES Laundry_Service(service_id));



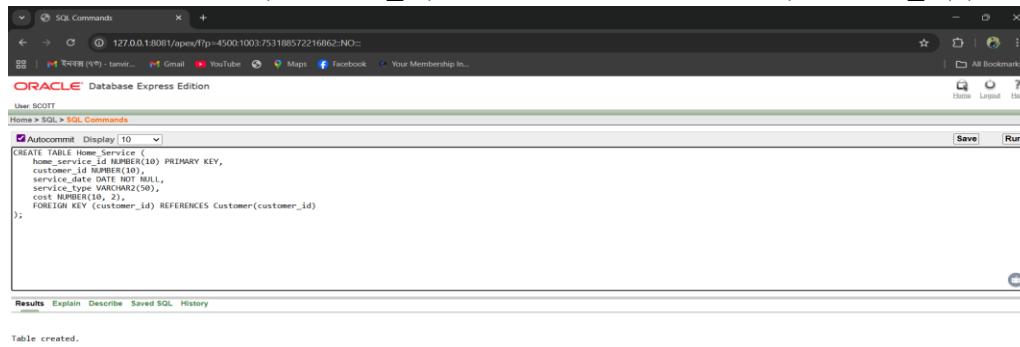
```
CREATE TABLE Owner_Laundry_Service (
  owner_id NUMBER(10),
  service_id NUMBER(10),
  PRIMARY KEY (owner_id, service_id),
  FOREIGN KEY (owner_id) REFERENCES Owner(owner_id),
  FOREIGN KEY (service_id) REFERENCES Laundry_Service(service_id)
);
```

Table created.
0.02 seconds



10. Home Service

CREATE TABLE Home_Service (
 home_service_id NUMBER(10) PRIMARY KEY,
 customer_id NUMBER(10),
 service_date DATE NOT NULL,
 service_type VARCHAR2(50),
 cost NUMBER(10, 2),
 FOREIGN KEY (customer_id) REFERENCES Customer(customer_id));



```
CREATE TABLE Home_Service (
  home_service_id NUMBER(10) PRIMARY KEY,
  customer_id NUMBER(10),
  service_date DATE NOT NULL,
  service_type VARCHAR2(50),
  cost NUMBER(10, 2),
  FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);
```

Table created.
0.00 seconds



Data Insertion Using SQL:

1.Customer :

INSERT INTO Customer (customer_id, first_name, last_name, phone_number, email, address)
 VALUES (1, 'John', 'Doe', '555-1234', 'john.doe@example.com', '123 Main St, Anytown');

```
INSERT INTO Customer (customer_id, first_name, last_name, phone_number, email, address)
VALUES (2, 'Jane', 'Smith', '555-5678', 'jane.smith@example.com', '456 Oak Ave, Anytown');
INSERT INTO Customer (customer_id, first_name, last_name, phone_number, email, address)
VALUES (3, 'Peter', 'Jones', '555-9012', 'peter.jones@example.com', '789 Pine Ln, Anytown');
INSERT INTO Customer (customer_id, first_name, last_name, phone_number, email, address)
VALUES (4, 'Mary', 'Brown', '555-3456', 'mary.brown@example.com', '101 Maple Blvd, Anytown');
INSERT INTO Customer (customer_id, first_name, last_name, phone_number, email, address)
VALUES (5, 'Robert', 'Davis', '555-7890', 'robert.davis@example.com', '202 Birch Dr, Anytown');
```

The screenshot shows a browser window for Oracle Database Express Edition. The URL is 127.0.0.1:8081/apex/f?p=4500:1003:753188572216862::NO:::. The page title is "SQL Commands". The SQL command entered is "Select * from Customer". The results table has columns: CUSTOMER_ID, FIRST_NAME, LAST_NAME, PHONE_NUMBER, EMAIL, and ADDRESS. The data is as follows:

CUSTOMER_ID	FIRST_NAME	LAST_NAME	PHONE_NUMBER	EMAIL	ADDRESS
1	John	Doe	555-1234	john.doe@example.com	123 Main St, Anytown
2	Jane	Smith	555-5678	jane.smith@example.com	456 Oak Ave, Anytown
3	Peter	Jones	555-9012	peter.jones@example.com	789 Pine Ln, Anytown
4	Mary	Brown	555-3456	mary.brown@example.com	101 Maple Blvd, Anytown
5	Robert	Davis	555-7890	robert.davis@example.com	202 Birch Dr, Anytown

5 rows returned in 0.02 seconds. CSV Export button is present.

At the bottom, it says "Language: en-US" and "Application Express 2.1.0.0.39 Copyright © 1999, 2005, Oracle. All rights reserved."

2. Laundry Service:

```
INSERT INTO Laundry_Service (service_id, service_name, description, price) VALUES (1, 'Wash & Fold', 'Standard wash, dry, and fold service.', 1.50);
INSERT INTO Laundry_Service (service_id, service_name, description, price) VALUES (2, 'Dry Cleaning', 'Professional dry cleaning for delicate items.', 5.75);
INSERT INTO Laundry_Service (service_id, service_name, description, price) VALUES (3, 'Ironing', 'Ironing service, priced per garment.', 2.00);
INSERT INTO Laundry_Service (service_id, service_name, description, price) VALUES (4, 'Comforter Wash', 'Specialized wash for comforters and large blankets.', 15.00);
INSERT INTO Laundry_Service (service_id, service_name, description, price) VALUES (5, 'Express Service', 'Same-day service with a quick turnaround.', 2.50);
```

The screenshot shows the Oracle Database Express Edition interface. A SQL command window displays the following query:

```
Select * From Laundry_Service
```

The results show the following data:

SERVICE_ID	SERVICE_NAME	DESCRIPTION	PRICE
1	Wash & Fold	Standard wash, dry and fold service.	1.5
2	Dry Cleaning	Professional cleaning of delicate items.	3.75
3	Ironing	Ironing service, priced per garment.	2
4	Comforter Wash	Specialized wash for comforters and large blankets.	15
5	Express Service	Same-day service with a quick turnaround.	2.5

5 rows returned in 0.00 seconds

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



3. Employee :

INSERT INTO Employee (employee_id, first_name, last_name, job_title, hire_date, salary)
VALUES (1, 'Mark', 'Wilson', TO_DATE('2022-01-15', 'YYYY-MM-DD'), 60000.00);
INSERT INTO Employee (employee_id, first_name, last_name, job_title, hire_date, salary)
VALUES (2, 'Lisa', 'Taylor', 'Attendant', TO_DATE('2023-03-01', 'YYYY-MM-DD'), 35000.00);
INSERT INTO Employee (employee_id, first_name, last_name, job_title, hire_date, salary)
VALUES (3, 'Kevin', 'Harris', 'Dry Cleaner', TO_DATE('2022-05-20', 'YYYY-MM-DD'),
45000.00);

INSERT INTO Employee (employee_id, first_name, last_name, job_title, hire_date, salary)
VALUES (4, 'Emily', 'Clark', 'Customer Service Rep', TO_DATE('2023-08-10', 'YYYY-MM-DD'),
32000.00);

INSERT INTO Employee (employee_id, first_name, last_name, job_title, hire_date, salary)
VALUES (5, 'David', 'Miller', 'Delivery Driver', TO_DATE('2024-02-25', 'YYYY-MM-DD'),
38000.00);

The screenshot shows the Oracle Database Express Edition interface. A SQL command window displays the following query:

```
Select * From Employee
```

The results show the following data:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	JOB_TITLE	HIRE_DATE	SALARY
1	Mark	Wilson	Manager	15-JAN-22	60000
2	Lisa	Taylor	Attendant	01-MAR-23	35000
3	Kevin	Harris	Dry Cleaner	20-MAY-22	45000
4	Emily	Clark	Customer Service Rep	10-AUG-23	32000
5	David	Miller	Delivery Driver	25-FEB-24	38000

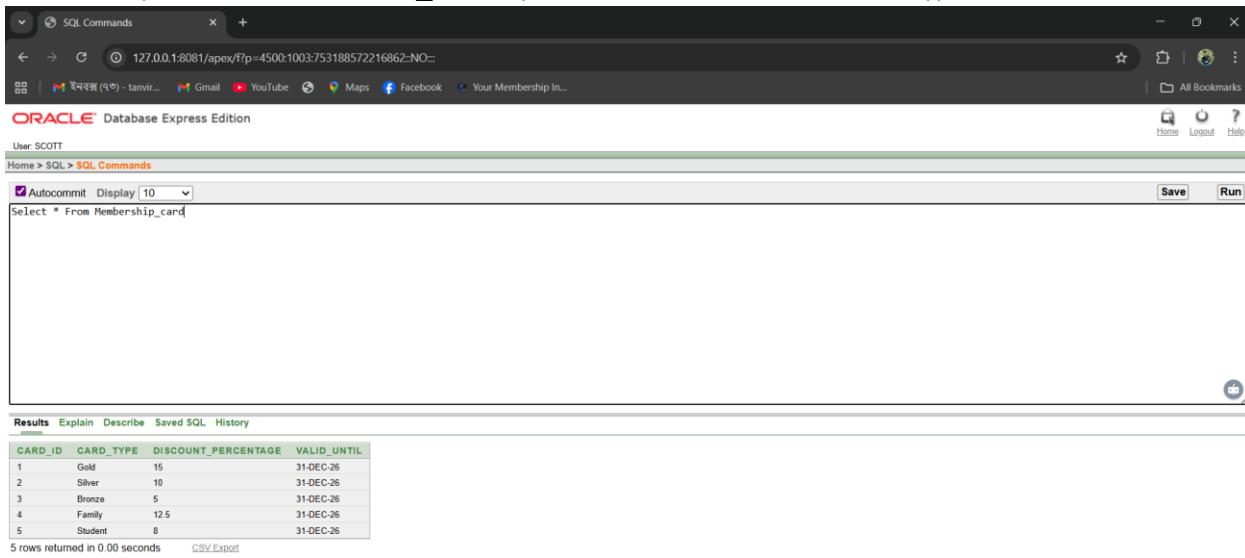
5 rows returned in 0.00 seconds

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



4.Membership Card :

```
INSERT INTO Membership_Card (card_id, card_type, discount_percentage, valid_until)
VALUES (1, 'Gold', 15.00, TO_DATE('2026-12-31', 'YYYY-MM-DD'));
INSERT INTO Membership_Card (card_id, card_type, discount_percentage, valid_until)
VALUES (2, 'Silver', 10.00, TO_DATE('2026-12-31', 'YYYY-MM-DD'));
INSERT INTO Membership_Card (card_id, card_type, discount_percentage, valid_until)
VALUES (3, 'Bronze', 5.00, TO_DATE('2026-12-31', 'YYYY-MM-DD'));
INSERT INTO Membership_Card (card_id, card_type, discount_percentage, valid_until)
VALUES (4, 'Family', 12.50, TO_DATE('2026-12-31', 'YYYY-MM-DD'));
INSERT INTO Membership_Card (card_id, card_type, discount_percentage, valid_until)
VALUES (5, 'Student', 8.00, TO_DATE('2026-12-31', 'YYYY-MM-DD'));
```



The screenshot shows the Oracle Database Express Edition interface. The user is executing a SQL command to select all rows from the Membership_Card table. The results are displayed in a grid:

CARD_ID	CARD_TYPE	DISCOUNT_PERCENTAGE	VALID_UNTIL
1	Gold	15	31-DEC-26
2	Silver	10	31-DEC-26
3	Bronze	5	31-DEC-26
4	Family	12.5	31-DEC-26
5	Student	8	31-DEC-26

5 rows returned in 0.00 seconds [CSV Export](#)

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



5.Customer Membership:

```
INSERT INTO Customer_Membership (customer_id, card_id, join_date) VALUES (1, 1,
TO_DATE('2024-01-10', 'YYYY-MM-DD'));
INSERT INTO Customer_Membership (customer_id, card_id, join_date) VALUES (2, 2,
TO_DATE('2024-02-15', 'YYYY-MM-DD'));
INSERT INTO Customer_Membership (customer_id, card_id, join_date) VALUES (3, 3,
TO_DATE('2024-03-20', 'YYYY-MM-DD'));
INSERT INTO Customer_Membership (customer_id, card_id, join_date) VALUES (4, 4,
TO_DATE('2024-04-25', 'YYYY-MM-DD'));
INSERT INTO Customer_Membership (customer_id, card_id, join_date) VALUES (5, 5,
TO_DATE('2024-05-30', 'YYYY-MM-DD'));
```

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

```
select * From Customer_Membership;
```

The results pane displays the following data:

CUSTOMER_ID	CARD_ID	JOIN_DATE
1	1	10-JAN-24
2	2	15-MAR-24
3	3	28-MAR-24
4	4	29-APR-24
5	5	30-MAY-24

5 rows returned in 0.00 seconds [CSV Export](#)

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

6.Utilities Bill:

INSERT INTO Utilities_Bill (bill_id, bill_type, amount, due_date, payment_status) VALUES (1, 'Electricity', 250.75, TO_DATE('2024-06-01', 'YYYY-MM-DD'), 'Paid');
 INSERT INTO Utilities_Bill (bill_id, bill_type, amount, due_date, payment_status) VALUES (2, 'Water', 85.50, TO_DATE('2024-06-10', 'YYYY-MM-DD'), 'Unpaid');
 INSERT INTO Utilities_Bill (bill_id, bill_type, amount, due_date, payment_status) VALUES (3, 'Gas', 120.00, TO_DATE('2024-06-15', 'YYYY-MM-DD'), 'Paid');
 INSERT INTO Utilities_Bill (bill_id, bill_type, amount, due_date, payment_status) VALUES (4, 'Internet', 75.00, TO_DATE('2024-06-20', 'YYYY-MM-DD'), 'Unpaid');
 INSERT INTO Utilities_Bill (bill_id, bill_type, amount, due_date, payment_status) VALUES (5, 'Electricity', 270.50, TO_DATE('2024-07-01', 'YYYY-MM-DD'), 'Unpaid');

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

```
Select * From Utilities_Bill;
```

The results pane displays the following data:

BILL_ID	BILL_TYPE	AMOUNT	DUEDATE	PAYMENT_STATUS
1	Electricity	250.75	01-JUN-24	Paid
2	Water	85.5	10-JUN-24	Unpaid
3	Gas	120	15-JUN-24	Paid
4	Internet	75	20-JUN-24	Unpaid
5	Electricity	270.5	01-JUL-24	Unpaid

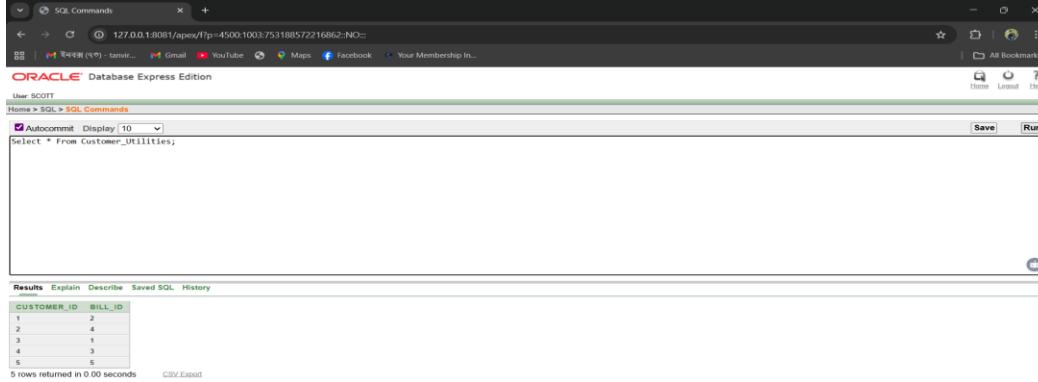
5 rows returned in 0.02 seconds [CSV Export](#)

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

7.Customer Utilities:

INSERT INTO Customer_Utils (customer_id, bill_id) VALUES (1, 2);

INSERT INTO Customer_Utils (customer_id, bill_id) VALUES (2, 4);
 INSERT INTO Customer_Utils (customer_id, bill_id) VALUES (3, 1);
 INSERT INTO Customer_Utils (customer_id, bill_id) VALUES (4, 3);
 INSERT INTO Customer_Utils (customer_id, bill_id) VALUES (5, 5);



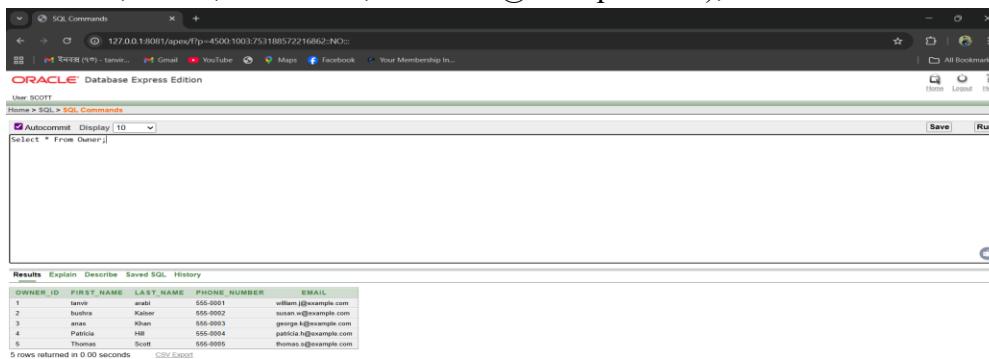
```
SQL Commands
User SCOTT
Home > SQL > SQL Commands
Select * From Customer_Utils;

Results Explain Describe Saved SQL History
CUSTOMER_ID BILL_ID
1 2
2 4
3 1
4 3
5 5
5 rows returned in 0.00 seconds
CSV Export
```



8.Owner:

INSERT INTO Owner (owner_id, first_name, last_name, phone_number, email) VALUES (1, 'William', 'Jackson', '555-0001', 'william.j@example.com');
 INSERT INTO Owner (owner_id, first_name, last_name, phone_number, email) VALUES (2, 'Susan', 'Wright', '555-0002', 'susan.w@example.com');
 INSERT INTO Owner (owner_id, first_name, last_name, phone_number, email) VALUES (3, 'George', 'King', '555-0003', 'george.k@example.com');
 INSERT INTO Owner (owner_id, first_name, last_name, phone_number, email) VALUES (4, 'Patricia', 'Hill', '555-0004', 'patricia.h@example.com');
 INSERT INTO Owner (owner_id, first_name, last_name, phone_number, email) VALUES (5, 'Thomas', 'Scott', '555-0005', 'thomas.s@example.com');



```
SQL Commands
User SCOTT
Home > SQL > SQL Commands
Select * From Owner;

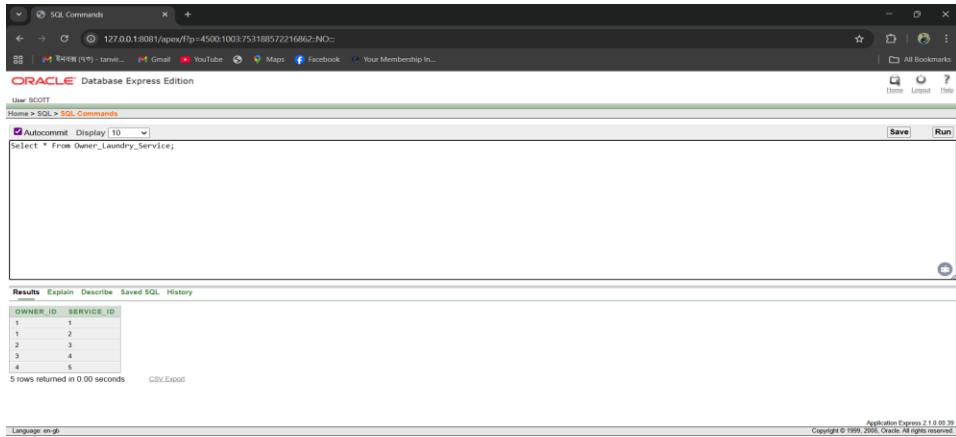
Results Explain Describe Saved SQL History
OWNER_ID FIRST_NAME LAST_NAME PHONE_NUMBER EMAIL
1 William Jackson 555-0001 william.j@example.com
2 Susan Wright 555-0002 susan.w@example.com
3 George King 555-0003 george.k@example.com
4 Patricia Hill 555-0004 patricia.h@example.com
5 Thomas Scott 555-0005 thomas.s@example.com
5 rows returned in 0.00 seconds
CSV Export
```



9.Owner Laundry Service :

INSERT INTO Owner_Laundry_Service (owner_id, service_id) VALUES (1, 1);
 INSERT INTO Owner_Laundry_Service (owner_id, service_id) VALUES (1, 2);

INSERT INTO Owner_Laundry_Service (owner_id, service_id) VALUES (2, 3);
 INSERT INTO Owner_Laundry_Service (owner_id, service_id) VALUES (3, 4);
 INSERT INTO Owner_Laundry_Service (owner_id, service_id) VALUES (4, 5);

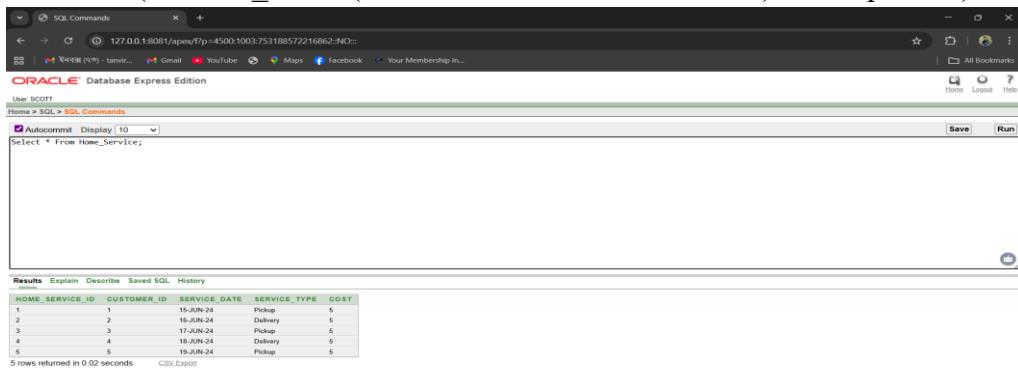


OWNER_ID	SERVICE_ID
1	1
1	2
2	3
3	4
4	5

5 rows returned in 0.00 seconds CSV Export

10.Home Service:

INSERT INTO Home_Service (home_service_id, customer_id, service_date, service_type, cost)
 VALUES (1, 1, TO_DATE('2024-06-15', 'YYYY-MM-DD'), 'Pickup', 5.00);
 INSERT INTO Home_Service (home_service_id, customer_id, service_date, service_type, cost)
 VALUES (2, 2, TO_DATE('2024-06-16', 'YYYY-MM-DD'), 'Delivery', 5.00);
 INSERT INTO Home_Service (home_service_id, customer_id, service_date, service_type, cost)
 VALUES (3, 3, TO_DATE('2024-06-17', 'YYYY-MM-DD'), 'Pickup', 5.00);
 INSERT INTO Home_Service (home_service_id, customer_id, service_date, service_type, cost)
 VALUES (4, 4, TO_DATE('2024-06-18', 'YYYY-MM-DD'), 'Delivery', 5.00);
 INSERT INTO Home_Service (home_service_id, customer_id, service_date, service_type, cost)
 VALUES (5, 5, TO_DATE('2024-06-19', 'YYYY-MM-DD'), 'Pickup', 5.00);



HOME_SERVICE_ID	CUSTOMER_ID	SERVICE_DATE	SERVICE_TYPE	COST
1	1	15-JUN-24	Pickup	5
2	2	16-JUN-24	Delivery	5
3	3	17-JUN-24	Pickup	5
4	4	18-JUN-24	Delivery	5
5	5	19-JUN-24	Pickup	5

5 rows returned in 0.02 seconds CSV Export



Missing basic PL/SQL query (Variables, operators, single-row-function, group-function)

Variables:

1. Names and data types of the two variables declared.

```

DECLARE
  v_product_id  NUMBER(5);
  v_product_name VARCHAR2(100);
BEGIN
  v_product_id := 250;
  v_product_name := 'Wireless Keyboard';
  DBMS_OUTPUT.PUT_LINE('Product ID: ' || v_product_id);
  DBMS_OUTPUT.PUT_LINE('Product Name: ' || v_product_name);
END;
/

```

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

```

DECLARE
  v_product_id  NUMBER(5);
  v_product_name VARCHAR2(100);
BEGIN
  v_product_id := 250;
  v_product_name := 'Wireless Keyboard';
  DBMS_OUTPUT.PUT_LINE('Product ID: ' || v_product_id);
  DBMS_OUTPUT.PUT_LINE('Product Name: ' || v_product_name);
END;
/

```

The results pane displays the output:

```

Product ID: 250
Product Name: Wireless Keyboard
Statement processed.

0.08 seconds

```

At the bottom, the status bar indicates "Application Express 2.1.0.0.39" and the date "9/16/2025".

2. In the provided PL/SQL block, Find value of v_product_id and data type of the v_product_name variable.

```

DECLARE
  v_employee_id employees.employee_id%TYPE := 101;
  v_employee_name employees.first_name%TYPE;
BEGIN
  SELECT first_name
  INTO v_employee_name
  FROM employees
  WHERE employee_id = v_employee_id;
  DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_employee_name);
END;
/

```

```

DECLARE
  v_product_id NUMBER := 10;
  v_product_name VARCHAR2(50) := 'Laptop';
  v_is_available BOOLEAN := TRUE;
BEGIN
  -- Display the variables
  DBMS_OUTPUT.PUT_LINE('Product ID: ' || v_product_id);
  DBMS_OUTPUT.PUT_LINE('Product Name: ' || v_product_name);

  -- Use a variable in a conditional statement
  IF v_is_available THEN
    DBMS_OUTPUT.PUT_LINE('Status: Available for sale.');
  END IF;
END;
/

```

Product ID: 10
Product Name: Laptop
Status: Available for sale.
Statement processed.
0.00 seconds

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

Operators:

1. In the provided PL/SQL block, two Find the final value assigned to the v_total_pay variable after the calculation is complete.

DECLARE

v_base_salary NUMBER := 50000;

v_bonus_rate NUMBER := 0.10;

v_total_pay NUMBER;

BEGIN

v_total_pay := v_base_salary + (v_base_salary * v_bonus_rate);

DBMS_OUTPUT.PUT_LINE('Total Annual Pay: ' || v_total_pay);

END;

/

```

DECLARE
  v_base_salary NUMBER := 50000;
  v_bonus_rate NUMBER := 0.10;
  v_total_pay NUMBER;
BEGIN
  v_total_pay := v_base_salary + (v_base_salary * v_bonus_rate);
  DBMS_OUTPUT.PUT_LINE('Total Annual Pay: ' || v_total_pay);
END;
/

```

Total Annual Pay: 55000
Statement processed.
0.00 seconds

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



2. Write logical operator is used in the IF statement, and the final output.

DECLARE

```
v_score1 NUMBER := 90;
v_score2 NUMBER := 85;
v_status VARCHAR2(20);

BEGIN
IF v_score1 >= 85 AND v_score2 >= 85 THEN
    v_status := 'Passed with Distinction';
ELSE
    v_status := 'Passed';
END IF;
DBMS_OUTPUT.PUT_LINE('Exam Status: ' || v_status);
END;
```

/

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the code provided above is executed. The results show the output: "Exam Status: Passed with Distinction". Below the results, it says "Statement processed." and "0.00 seconds". At the bottom, the status bar indicates "Language en-gb" and "Application Express 2.1.0.0.39 Copyright © 1995, 2006, Oracle. All rights reserved." The system tray at the bottom of the screen shows the date and time as 9/16/2025 7:34 PM.

Single-row-function

1. Find the value is stored in the v_initials variable after the block is executed.

DECLARE

```
v_full_name VARCHAR2(100) := 'john smith';
v_initials VARCHAR2(10);

BEGIN
v_full_name := INITCAP(v_full_name);
v_initials := SUBSTR(v_full_name, 1, 1) || '.' || SUBSTR(v_full_name, INSTR(v_full_name, ' ') + 1, 1);
DBMS_OUTPUT.PUT_LINE('Formatted Name: ' || v_full_name);
DBMS_OUTPUT.PUT_LINE('Initials: ' || v_initials);
END;
```

/

```

DECLARE
  v_full_name VARCHAR2(100);
  v_initials VARCHAR2(10);
BEGIN
  v_full_name := INITCAP(v_full_name);
  v_initials := SUBSTR(v_full_name, 1, 1) || '.' || SUBSTR(v_full_name, INSTR(v_full_name, ' ') + 1, 1);
  DBMS_OUTPUT.PUT_LINE('Formatted Name: ' || v_full_name);
  DBMS_OUTPUT.PUT_LINE('Initials: ' || v_initials);
END;
/

```

Results Explain Describe Saved SQL History

Formatted Name: John Smith
Initials: J.S
Statement processed.

0.01 seconds



2. Name of the function used to get the current date and display of v_invoice_total.

DECLARE

```

v_current_date DATE := SYSDATE;
v_invoice_total NUMBER := 1250.75;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Current Date: ' || TO_CHAR(v_current_date, 'DD-MON-YYYY'));
  DBMS_OUTPUT.PUT_LINE('Invoice Amount: ' || TO_CHAR(v_invoice_total, '$9,999.00'));
END;
/

```

```

DECLARE
  v_current_date DATE := SYSDATE;
  v_invoice_total NUMBER := 1250.75;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Current Date: ' || TO_CHAR(v_current_date, 'DD-MON-YYYY'));
  DBMS_OUTPUT.PUT_LINE('Invoice Amount: ' || TO_CHAR(v_invoice_total, '$9,999.00'));
END;
/

```

Results Explain Describe Saved SQL History

Current Date: 16-SEP-2025
Invoice Amount: \$1,250.75
Statement processed.

0.00 seconds



Group-function

- Find the COUNT (item_cost) function and return a different value than the SUM(item_cost) function.

DECLARE

```

v_num_items NUMBER;
v_total_cost NUMBER;
BEGIN

```

```

SELECT COUNT(item_cost), SUM(item_cost)
INTO v_num_items, v_total_cost
FROM (SELECT 100 AS item_cost FROM DUAL UNION ALL
      SELECT 200 AS item_cost FROM DUAL UNION ALL
      SELECT 150 AS item_cost FROM DUAL);
DBMS_OUTPUT.PUT_LINE('Total number of items: ' || v_num_items);
DBMS_OUTPUT.PUT_LINE('Total cost of items: ' || v_total_cost);
END; /

```

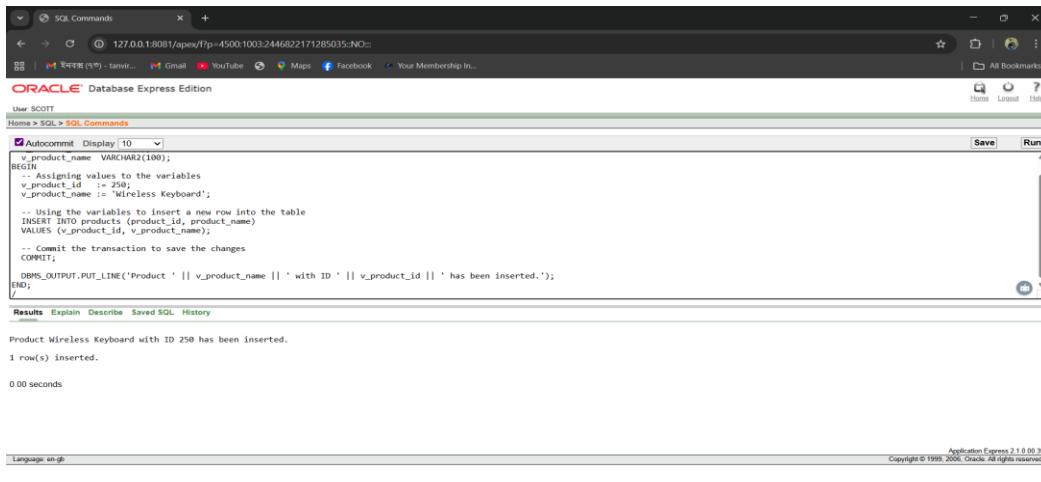
The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a PL/SQL block is run. The code declares two variables, v_num_items and v_total_cost, both of type NUMBER. It then uses a subquery to select COUNT(item_cost) and SUM(item_cost) from a union of three constant rows (100, 200, and 150). Finally, it outputs the total number of items and the total cost using DBMS_OUTPUT.PUT_LINE. The output pane shows the results: Total number of items: 3 and Total cost of items: 450.

2. In this PL/SQL block, Find the role of the v_product_id and v_product_name variables.

```

DECLARE
v_product_id  NUMBER(5);
v_product_name VARCHAR2(100);
BEGIN
v_product_id := 250;
v_product_name := 'Wireless Keyboard';
INSERT INTO products (product_id, product_name)
VALUES (v_product_id, v_product_name);
COMMIT;
DBMS_OUTPUT.PUT_LINE('Product ' || v_product_name || ' with ID ' || v_product_id || ' has
been inserted.');
END;
/

```



```

SQL Commands
127.0.0.1:8081/apex/?p=4500:1003:244682171285035:NO:_
ORACLE Database Express Edition
User SCOTT
Home > SQL > SQL Commands
 Autocommit Display 10
v_product_name VARCHAR(100);
BEGIN
    -- Assigning values to the variables
    v_product_id := 250;
    v_product_name := 'Product Wireless Keyboard';
    -- Using the variables to insert a new row into the table
    INSERT INTO products (product_id, product_name)
    VALUES (v_product_id, v_product_name);
    -- Commit the transaction to save the changes
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Product ' || v_product_name || ' with ID ' || v_product_id || ' has been inserted.');
END;
/

```

Results Explain Describe Saved SQL History

Product Wireless Keyboard with ID 250 has been inserted.
1 row(s) inserted.

0.00 seconds

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

Loop

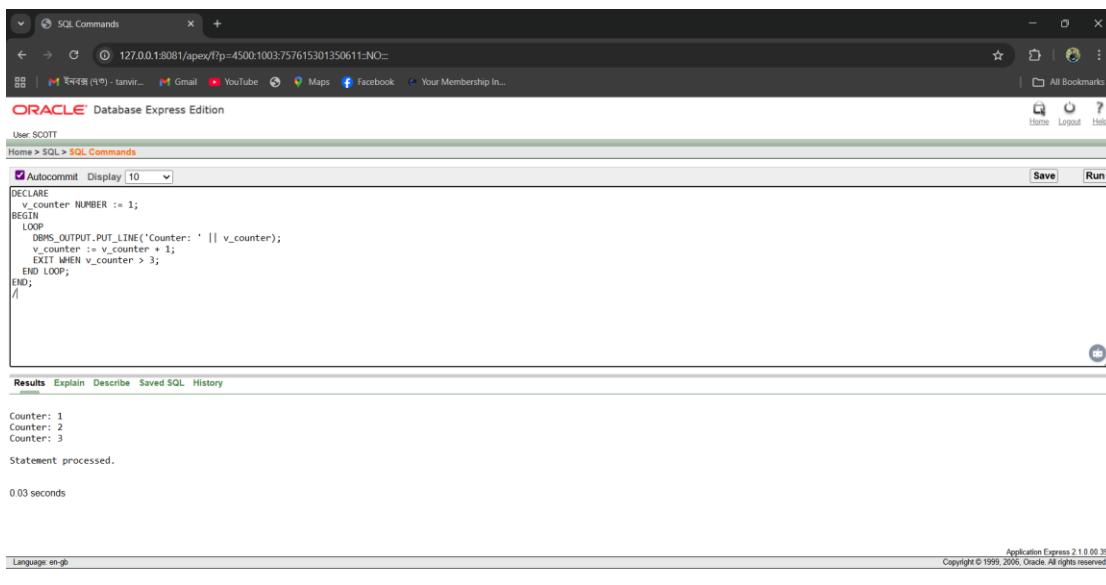
1. Write a PL/SQL block that uses a WHILE loop to print the numbers from 3 down to 1.

DECLARE

```

v_counter NUMBER := 1;
BEGIN
LOOP
    DBMS_OUTPUT.PUT_LINE('Counter: ' || v_counter);
    v_counter := v_counter + 1;
    EXIT WHEN v_counter > 3;
END LOOP;
END;
/

```



```

SQL Commands
127.0.0.1:8081/apex/?p=4500:1003:757615301350611:NO:_
ORACLE Database Express Edition
User SCOTT
Home > SQL > SQL Commands
 Autocommit Display 10
DECLARE
    v_counter NUMBER := 1;
BEGIN
    LOOP
        DBMS_OUTPUT.PUT_LINE('Counter: ' || v_counter);
        v_counter := v_counter + 1;
        EXIT WHEN v_counter > 3;
    END LOOP;
END;
/

```

Results Explain Describe Saved SQL History

Counter: 1
Counter: 2
Counter: 3

Statement processed.

0.03 seconds

Language: en-gb 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 print the numbers from 5 down to 1.

```

DECLARE
    v_counter NUMBER := 1;
BEGIN
    LOOP
        DBMS_OUTPUT.PUT_LINE('Counter: ' || v_counter);
        v_counter := v_counter + 1;
        EXIT WHEN v_counter > 5;
    END LOOP;
END;
/

```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the following PL/SQL block is run:

```

DECLARE
    v_counter NUMBER := 1;
BEGIN
    LOOP
        DBMS_OUTPUT.PUT_LINE('Counter: ' || v_counter);
        v_counter := v_counter + 1;
        EXIT WHEN v_counter > 5;
    END LOOP;
END;
/

```

The results pane displays the output of the loop:

```

Counter: 1
Counter: 2
Counter: 3
Counter: 4
Counter: 5
Statement processed.
0.04 seconds

```

Conditional Statement

1. Write a PL/SQL block that iterates through the employees table, calculates a bonus for each employee based on their job, and updates a separate bonuses table. The bonus amount is 10% of the salary for IT_PROG jobs and 5% for all other jobs.

```

DECLARE
    CURSOR c_employees IS
        SELECT employee_id, job_id, salary
        FROM employees e
        WHERE e.department_id = (SELECT department_id FROM departments WHERE department_name = 'IT');
    v_bonus_amount NUMBER;
BEGIN
    FOR emp_rec IN (SELECT e.employee_id, e.job_id, e.salary
                    FROM employees e
                    JOIN departments d ON e.department_id = d.department_id
                    WHERE d.department_name = 'Shipping')
    LOOP
        IF emp_rec.job_id = 'IT_PROG' THEN
            v_bonus_amount := emp_rec.salary * 0.10;
        ELSE
            v_bonus_amount := emp_rec.salary * 0.05;
        END IF;
        -- Insert logic to update the bonuses table
    END LOOP;
END;
/

```

```

ELSE
  v_bonus_amount := emp_rec.salary * 0.05;
END IF;
IF NOT EXISTS (SELECT 1 FROM bonuses WHERE employee_id = emp_rec.employee_id)
THEN
  INSERT INTO bonuses (employee_id, bonus_date, bonus_amount)
  VALUES (emp_rec.employee_id, SYSDATE, v_bonus_amount);
END IF;
END LOOP;
COMMIT;
END;
/

```



The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a PL/SQL block is run. The code declares variables for hours worked (45), hourly rate (20), and paycheck. It then calculates the paycheck based on whether hours worked are greater than 40. The output shows the total paycheck is \$950.

2. Write a PL/SQL block to assign a letter grade to a student based on their score. The grading scale is: 'A' for scores 90-100, 'B' for 80-89, 'C' for 70-79, and 'F' for any score below 70.

```

DECLARE
  v_student_score NUMBER := 85;
  v_grade      VARCHAR2(1);
BEGIN
  IF v_student_score >= 90 THEN
    v_grade := 'A';
  ELSIF v_student_score >= 80 THEN
    v_grade := 'B';
  ELSIF v_student_score >= 70 THEN
    v_grade := 'C';
  ELSE
    v_grade := 'F';
  
```

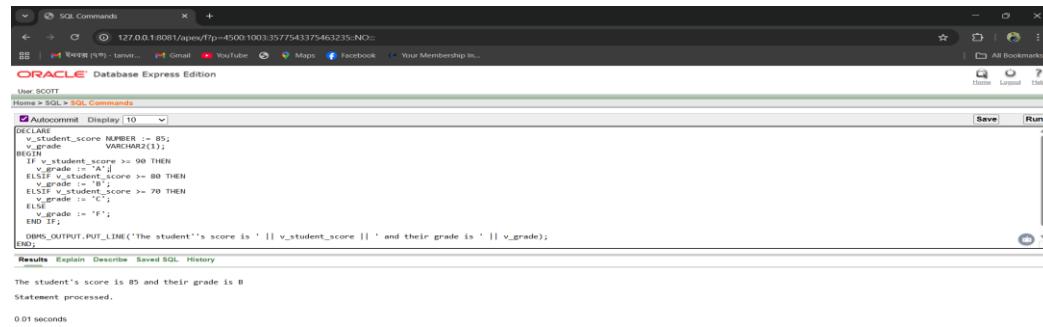
Language: en-US Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved. 2:24 AM 9/17/2025

Rain coming In about 3 hours

```

END IF;
DBMS_OUTPUT.PUT_LINE('The student''s score is ' || v_student_score || ' and their grade is ' || v_grade);
END;
/

```



```

DECLARE
  v_student_score NUMBER := 85;
  v_grade VARCHAR2(1);
BEGIN
  IF v_student_score >= 90 THEN
    v_grade := 'A';
  ELSIF v_student_score >= 80 THEN
    v_grade := 'B';
  ELSIF v_student_score >= 70 THEN
    v_grade := 'C';
  ELSE
    v_grade := 'F';
  END IF;
  DBMS_OUTPUT.PUT_LINE('The student''s score is ' || v_student_score || ' and their grade is ' || v_grade);
END;
/

```

The student's score is 85 and their grade is B
Statement processed.
0.01 seconds



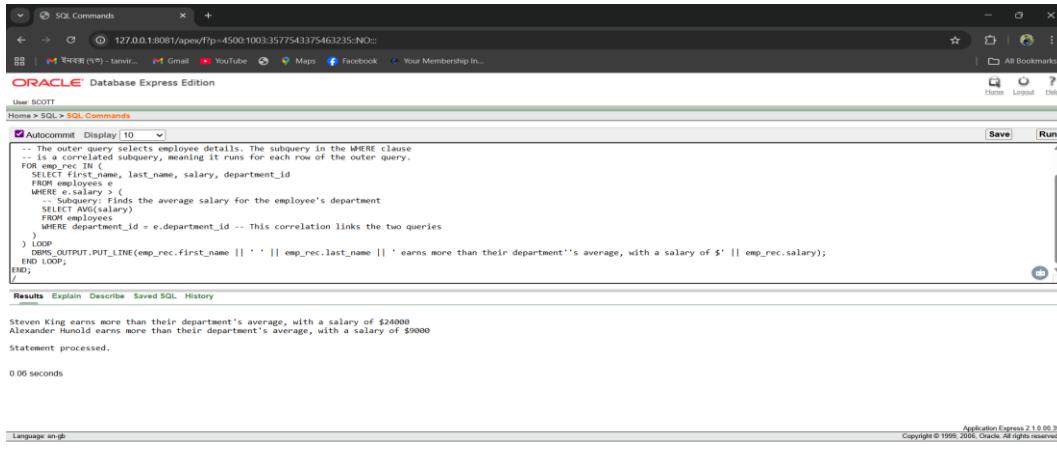
Subquery

1. Write a PL/SQL block that lists the name and salary of every employee who earns more than the average salary of their own department

```

DECLARE
  v_employee_name VARCHAR2(100);
  v_employee_salary NUMBER;
BEGIN
  FOR emp_rec IN (
    SELECT first_name, last_name, salary, department_id
    FROM employees e
    WHERE e.salary > (
      SELECT AVG(salary)
      FROM employees
      WHERE department_id = e.department_id -- This correlation links the two queries
    )
  ) LOOP
    DBMS_OUTPUT.PUT_LINE(emp_rec.first_name || ' ' || emp_rec.last_name || ' earns more than
their department''s average, with a salary of $' || emp_rec.salary);
  END LOOP;
END;
/

```



The screenshot shows the Oracle Database Express Edition SQL Commands interface. A PL/SQL block is being run under user SCOTT. The code uses a correlated subquery to find employees whose salary is higher than the average salary of their department. The results show two rows: Steven King and Alexander Hunold.

```

-- The outer query selects employee details. The subquery in the WHERE clause
-- is a correlated subquery, meaning it runs for each row of the outer query.
FOR emp_rec IN (
  SELECT first_name, last_name, salary, department_id
  FROM employees e
  WHERE e.salary > (
    -- Subquery: Finds the average salary for the employee's department
    SELECT AVG(salary)
    FROM employees
    WHERE department_id = e.department_id -- This correlation links the two queries
  )
) LOOP
  DBMS_OUTPUT.PUT_LINE(emp_rec.first_name || ' ' || emp_rec.last_name || ' earns more than their department''s average, with a salary of $' || emp_rec.salary);
END LOOP;
/

```

Results:

```

Steven King earns more than their department's average, with a salary of $24000
Alexander Hunold earns more than their department's average, with a salary of $90000

```

Statement processed.

0.06 seconds

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

2. Write a PL/SQL block to list the name and price of the most expensive product within each product category.

DECLARE

```
v_employee_name VARCHAR2(100);
v_employee_salary NUMBER;
```

BEGIN

```
FOR emp_rec IN (
  SELECT first_name, last_name, salary, department_id
  FROM employees e
  WHERE e.salary < (
    SELECT AVG(salary)
    FROM employees
    WHERE department_id = e.department_id -- This correlation links the two queries
  )
) LOOP
  DBMS_OUTPUT.PUT_LINE(emp_rec.first_name || ' ' || emp_rec.last_name || ' earns more than
their department''s average, with a salary of $' || emp_rec.salary);
END LOOP;
/

```

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

```

-- The outer query selects employee details. The subquery in the WHERE clause
-- uses a correlated subquery, meaning it runs for each row of the outer query.
FOR emp_rec IN (
  SELECT first_name, last_name, salary, department_id
  FROM employees
  WHERE salary < (
    -- Subquery: Finds the average salary for the employee's department
    SELECT AVG(salary)
    FROM employees
    WHERE department_id = e.department_id -- This correlation links the two queries
  ) LOOP
    DBMS_OUTPUT.PUT_LINE(emp_rec.first_name || ' ' || emp_rec.last_name || ' earns more than their department''s average, with a salary of $' || emp_rec.salary);
END LOOP;
END;
/

```

The output shows results for three employees:

Neena Kochhar earns more than their department's average, with a salary of \$17000
 Lex De Haan earns more than their department's average, with a salary of \$17000
 Bruce Ernst earns more than their department's average, with a salary of \$6000

Statement processed.

0.02 seconds

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

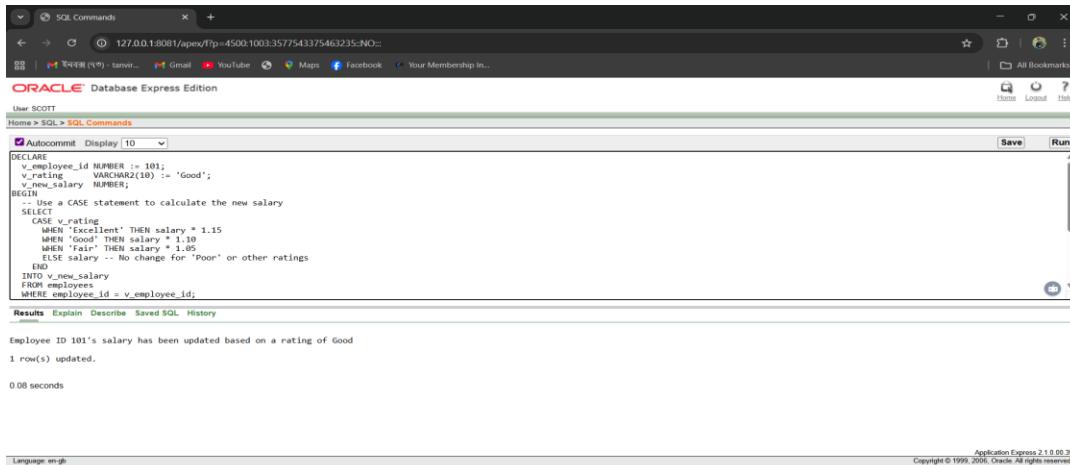
Joining

1. Write a PL/SQL block that uses a self-join to list every employee and the name of their direct manager. The output should include the employee's full name and their manager's full name.

```

DECLARE
  v_employee_id NUMBER := 101;
  v_rating      VARCHAR2(10) := 'Good';
  v_new_salary  NUMBER;
BEGIN
  SELECT
    CASE v_rating
      WHEN 'Excellent' THEN salary * 1.15
      WHEN 'Good' THEN salary * 1.10
      WHEN 'Fair' THEN salary * 1.05
      ELSE salary -- No change for 'Poor' or other ratings
    END
    INTO v_new_salary
    FROM employees
    WHERE employee_id = v_employee_id;
  UPDATE employees
    SET salary = v_new_salary
    WHERE employee_id = v_employee_id;
  DBMS_OUTPUT.PUT_LINE('Employee ID ' || v_employee_id || "'s salary has been updated
based on a rating of' || v_rating);
  COMMIT;
END;
/

```



The screenshot shows a browser window for Oracle Database Express Edition. The URL is 127.0.0.1:8081/apex/F?p=4500:1003:3577543375463235::NO:::. The page title is "SQL Commands". The user is SCOTT. The SQL code is:

```

DECLARE
    v_employee_id NUMBER := 101;
    v_rating      VARCHAR2(10) := 'Good';
    v_new_salary  NUMBER;
BEGIN
    -- Use a CASE statement to calculate the new salary
    SELECT
        CASE v_rating
        WHEN 'Excellent' THEN salary * 1.15
        WHEN 'Good' THEN salary * 1.10
        WHEN 'Fair' THEN salary * 1.05
        ELSE salary -- No change for 'Poor' or other ratings
        END
    INTO v_new_salary
    FROM employees
    WHERE employee_id = v_employee_id;

```

The results show: Employee ID 101's salary has been updated based on a rating of Good. 1 row(s) updated. 0.08 seconds.

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



2. Write a PL/SQL block that joins the employees and projects tables. The block should iterate through all employees and display their name and the name of the project they are assigned to. If an employee is not assigned a project, the output should indicate this.

DECLARE

```

v_employee_id NUMBER := 102;
v_rating      VARCHAR2(10) := 'Excellent';
v_new_salary  NUMBER;

```

BEGIN

SELECT

CASE v_rating

WHEN 'Excellent' THEN salary * 1.15

WHEN 'Good' THEN salary * 1.10

WHEN 'Fair' THEN salary * 1.05

ELSE salary -- No change for 'Poor' or other ratings

END

INTO v_new_salary

FROM employees

WHERE employee_id = v_employee_id;

UPDATE employees

SET salary = v_new_salary

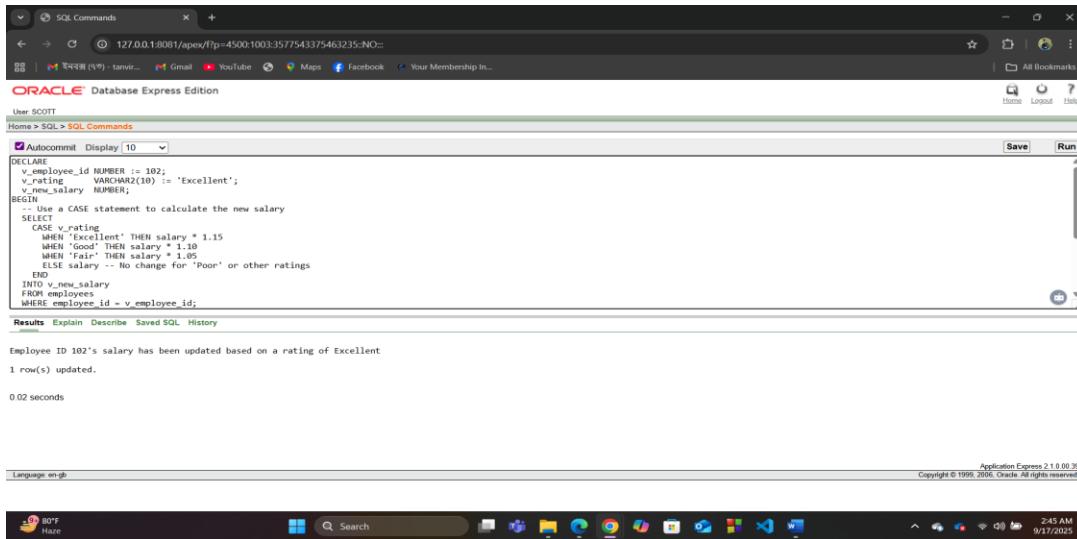
WHERE employee_id = v_employee_id;

DBMS_OUTPUT.PUT_LINE('Employee ID ' || v_employee_id || "'s salary has been updated based on a rating of' || v_rating);

COMMIT;

END;

/



The screenshot shows the Oracle Database Express Edition SQL Commands interface. A PL/SQL script is being run to update an employee's salary based on their rating. The script uses a CASE statement to calculate a 15% bonus for 'Excellent' rating. The output shows that one row was updated successfully.

```

DECLARE
  v_employee_id NUMBER := 102;
  v_rating      VARCHAR2(10) := 'Excellent';
  v_new_salary  NUMBER;
BEGIN
  -- Use a CASE statement to calculate the new salary
  SELECT v_new_salary
  CASE v_rating
    WHEN 'Excellent' THEN salary * 1.15
    WHEN 'Good'      THEN salary * 1.10
    WHEN 'Fair'      THEN salary * 1.05
    ELSE salary -- No change for 'Poor' or other ratings
  END
  INTO v_new_salary
  FROM employees
  WHERE employee_id = v_employee_id;
END;
/

```

Result:

```

Employee ID 102's salary has been updated based on a rating of Excellent
1 row(s) updated.

0.02 seconds

```

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

Advance PL/SQL(Missing Query)

Store function

1. A company wants a function to calculate an employee's total compensation, which is their salary plus a 15% bonus. The function should accept the employee's ID and return the total amount.

```

CREATE OR REPLACE FUNCTION fn_get_total_compensation (
  p_employee_id IN employees.employee_id%TYPE
)
RETURN NUMBER
AS
  v_salary employees.salary%TYPE;
BEGIN
  SELECT salary
  INTO v_salary
  FROM employees
  WHERE employee_id = p_employee_id;
  RETURN v_salary * 1.15;
END;
/

```

```

CREATE OR REPLACE FUNCTION fn_get_total_compensation (
    p_employee_id IN employees.employee_id%TYPE
)
RETURN NUMBER
AS
    v_salary employees.salary%TYPE;
BEGIN
    SELECT salary
    INTO v_salary
    FROM employees
    WHERE employee_id = p_employee_id;
    RETURN v_salary * 1.15;
END;
/

```

Function created.
0.03 seconds



2. Create a function that takes an employee's ID as input and returns their full name (first name and last name combined).

```

CREATE OR REPLACE FUNCTION fn_get_employee_name (
    p_employee_id IN employees.employee_id%TYPE
)
RETURN VARCHAR2
AS
    v_full_name VARCHAR2(100);
BEGIN
    SELECT first_name || ' ' || last_name
    INTO v_full_name
    FROM employees
    WHERE employee_id = p_employee_id;

    RETURN v_full_name;
END; /

```

```

CREATE OR REPLACE FUNCTION fn_get_employee_name (
    p_employee_id IN employees.employee_id%TYPE
)
RETURN VARCHAR2
AS
    v_full_name VARCHAR2(100);
BEGIN
    SELECT first_name || ' ' || last_name
    INTO v_full_name
    FROM employees
    WHERE employee_id = p_employee_id;
    RETURN v_full_name;
END;
/

```

Function created.
0.02 seconds

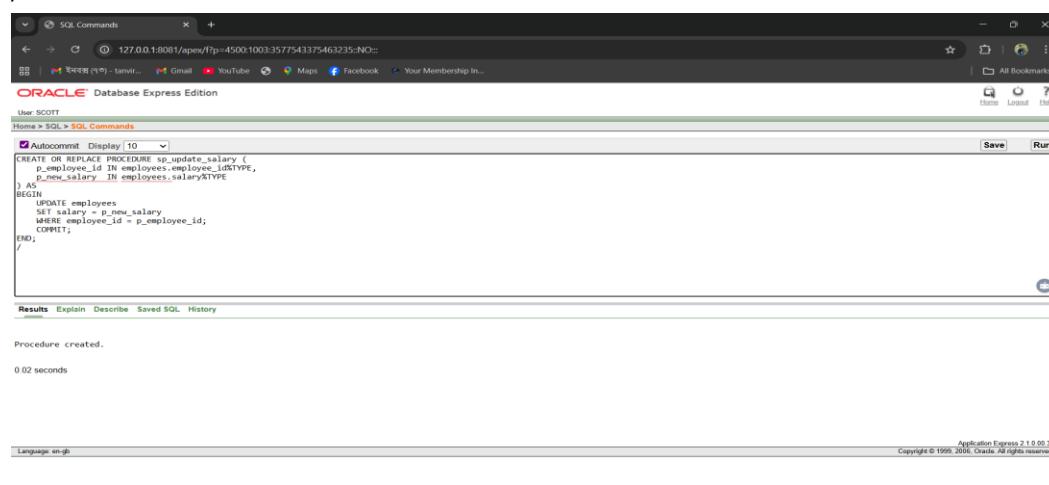


Store procedure

1. Write a stored procedure named sp_update_salary that accepts an employee's ID and a new salary, then updates the employee's record with the new salary.

```
CREATE OR REPLACE PROCEDURE sp_update_salary (
    p_employee_id IN employees.employee_id%TYPE,
    p_new_salary  IN employees.salary%TYPE
) AS
BEGIN
    UPDATE employees
    SET salary = p_new_salary
    WHERE employee_id = p_employee_id;
    COMMIT;
END;
```

/



The screenshot shows a browser window for Oracle Database Express Edition. The URL is 127.0.0.1:8081/apex/f?p=4500:1003:3577543375463235::NO:::. The page displays the SQL command for creating the procedure:

```
CREATE OR REPLACE PROCEDURE sp_update_salary (
    p_employee_id IN employees.employee_id%TYPE,
    p_new_salary  IN employees.salary%TYPE
) AS
BEGIN
    UPDATE employees
    SET salary = p_new_salary
    WHERE employee_id = p_employee_id;
    COMMIT;
END;
```

Below the code, it says "Procedure created." and "0.02 seconds". At the bottom, there is a status bar with "Language: en-gb" and "Application Express 2.1.0.0.39 Copyright © 1999-2006, Oracle. All rights reserved."

2. Write a procedure that accepts a name and displays a personalized greeting.

```
CREATE OR REPLACE PROCEDURE greet_person (
```

```
    p_name IN VARCHAR2
) AS
BEGIN
    DBMS_OUTPUT.PUT_LINE('Hello, ' || p_name || '! Welcome to the world of PL/SQL.');
END;
```

/

```

CREATE OR REPLACE PROCEDURE greet_person (
    p_name IN VARCHAR2
) AS
BEGIN
    DBMS_OUTPUT.PUT_LINE('Hello, ' || p_name || '! Welcome to the world of PL/SQL.');
END;
/

```

Procedure created.

0.02 seconds



Table Based record

1. Write a PL/SQL block that uses an explicit cursor in a FOR loop to list the names and salaries of all employees.

BEGIN

```

-- The FOR loop implicitly declares a cursor and a record variable
FOR emp_rec IN (SELECT first_name, last_name, salary FROM employees) LOOP
    DBMS_OUTPUT.PUT_LINE(
        'Employee: ' || emp_rec.first_name || ' ' || emp_rec.last_name ||
        ', Salary: ' || emp_rec.salary
    );
END LOOP;
END;
/

```

```

BEGIN
    -- The FOR loop implicitly declares a cursor and a record variable
    FOR emp_rec IN (SELECT first_name, last_name, salary FROM employees) LOOP
        DBMS_OUTPUT.PUT_LINE(
            'Employee: ' || emp_rec.first_name || ' ' || emp_rec.last_name ||
            ', Salary: ' || emp_rec.salary
        );
    END LOOP;
END;
/

```

Employee: Steven King, Salary: 29040
Employee: Neena Kochhar, Salary: 19550
Employee: Lex De Haan, Salary: 17000
Employee: Alexander Hunold, Salary: 9000
Employee: Bruce Ernst, Salary: 6000

Statement processed.

0.03 seconds

Language: en-gb Application Express 2.1.0.00.39 Copyright © 1999, 2006, Oracle. All rights reserved. 3:07 AM 9/17/2025



2. Write a PL/SQL block that retrieves all the data for one employee and stores it in a single record variable, then displays some of the information.

DECLARE

```
v_employee_rec employees%ROWTYPE;
BEGIN
  SELECT *
    INTO v_employee_rec
   FROM employees
  WHERE employee_id = 101;
  DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_employee_rec.first_name);
  DBMS_OUTPUT.PUT_LINE('Employee Salary: ' || v_employee_rec.salary);
END;
/
```

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

```
DECLARE
  This variable can hold an entire row from the employees table
  v_employee_rec employees%ROWTYPE;
BEGIN
  -- Fetch all columns for employee ID 101 into the record variable
  SELECT *
    INTO v_employee_rec
   FROM employees
  WHERE employee_id = 101;
  -- Access the data using dot notation
  DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_employee_rec.first_name);
  DBMS_OUTPUT.PUT_LINE('Employee Salary: ' || v_employee_rec.salary);
END;
/
```

The results pane shows the output:

```
Employee Name: Steven
Employee Salary: 29040
Statement processed.
```

Execution time: 0.02 seconds

At the bottom right, it says Application Express 2.1.0.02.39 Copyright © 1999-2005, Oracle. All rights reserved.

Explicit cursor

1. Write a PL/SQL block that uses an explicit cursor to display the names and salaries of employees who earn more than \$10,000.

DECLARE

```
CURSOR c_high_earners IS
  SELECT first_name, last_name, salary
    FROM employees
   WHERE salary > 10000;
  v_first_name employees.first_name%TYPE;
  v_last_name employees.last_name%TYPE;
  v_salary    employees.salary%TYPE;
BEGIN
```

```

OPEN c_high_earners;
LOOP
  FETCH c_high_earners INTO v_first_name, v_last_name, v_salary;
  EXIT WHEN c_high_earners%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE('Employee: ' || v_first_name || ' ' || v_last_name || ', Salary: ' ||
v_salary);
END LOOP;
CLOSE c_high_earners;
END;
/

```

```

SQL Commands
127.0.0.1:8081/apex/f?p=4500:1003:3577543375463235::NO::
ORACLE Database Express Edition
User SCOTT
Home > SQL > SQL Commands
Autocommit Display 10 Save Run
DECLARE
  -- Declare the explicit cursor
  CURSOR c_high_earners IS
    SELECT first_name, last_name, salary
    FROM employees
    WHERE salary > 10000;
  -- Declare variables to hold the fetched data
  v_first_name employees.first_name%TYPE;
  v_last_name employees.last_name%TYPE;
  v_salary   employees.salary%TYPE;
BEGIN
  -- Open the cursor
  OPEN c_high_earners;
  Results Explain Describe Saved SQL History
Employee: Steven King, Salary: 20840
Employee: Neena Kochhar, Salary: 19950
Employee: Lex De Haan, Salary: 17000
Statement processed.
0.03 seconds
Language: en_gb 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 display the names and salaries of employees who earn less than \$30,000

DECLARE

```

CURSOR c_high_earners IS
  SELECT first_name, last_name, salary
  FROM employees
  WHERE salary < 30000;
  v_first_name employees.first_name%TYPE;
  v_last_name employees.last_name%TYPE;
  v_salary   employees.salary%TYPE;
BEGIN
  OPEN c_high_earners;
  LOOP
    FETCH c_high_earners INTO v_first_name, v_last_name, v_salary;
    EXIT WHEN c_high_earners%NOTFOUND;

```

```

DBMS_OUTPUT.PUT_LINE('Employee: ' || v_first_name || ' ' || v_last_name || ', Salary: ' ||
v_salary);
END LOOP;
CLOSE c_high_earners;
END;
/

```

```

DECLARE
  -- Declare the explicit cursor
  CURSOR c_high_earners IS
    SELECT first_name, last_name, salary
    FROM employees
    WHERE salary > 30000;
  -- Declare variables to hold the fetched data
  v_first_name employees.first_name%TYPE;
  v_last_name employees.last_name%TYPE;
  v_salary   employees.salary%TYPE;
BEGIN
  -- Open the cursor
  OPEN c_high_earners;
  -- Loop to process each row
  LOOP
    -- Fetch the next row from the cursor
    FETCH c_high_earners INTO v_first_name, v_last_name, v_salary;
    -- Exit the loop if no more rows are found
    EXIT WHEN c_high_earners%NOTFOUND;
    -- Output the employee's name and salary
    DBMS_OUTPUT.PUT_LINE('Employee: ' || v_first_name || ' ' || v_last_name || ', Salary: ' ||
      v_salary);
  END LOOP;
  -- Close the cursor
  CLOSE c_high_earners;
END;
/

```

Statement processed.

0.02 seconds

Language: en_gb

Cursor based record

1. Write a PL/SQL block that uses a cursor FOR loop to display the names and salaries of all employees.

BEGIN

FOR emp_rec IN (SELECT first_name, last_name, salary FROM employees)

LOOP

```

DBMS_OUTPUT.PUT_LINE('Name: ' || emp_rec.first_name || ' ' || emp_rec.last_name || ', Salary: ' || emp_rec.salary);

```

END LOOP;

END;

/

```

BEGIN
  -- The FOR loop automatically creates a cursor and a cursor-based record (emp_rec)
  FOR emp_rec IN (SELECT first_name, last_name, salary FROM employees)
  LOOP
    -- Access the data using the record's fields
    DBMS_OUTPUT.PUT_LINE('Name: ' || emp_rec.first_name || ' ' || emp_rec.last_name || ', Salary: ' || emp_rec.salary);
  END LOOP;
END;
/

```

Statement processed.

0.02 seconds

Language: en_gb



2. Write a PL/SQL block that retrieves all the data for a specific customer with customer_id 1 and stores it in a single record variable. Display the customer's name and email from the record.

DECLARE

```
v_customer_rec customer%ROWTYPE;
BEGIN
  SELECT *
    INTO v_customer_rec
   FROM customer
  WHERE customer_id = 1;
  DBMS_OUTPUT.PUT_LINE('Customer Name: ' || v_customer_rec.first_name || ' ' || v_customer_rec.last_name);
  DBMS_OUTPUT.PUT_LINE('Customer Email: ' || v_customer_rec.email);
END;
/
```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the code is executed, and the results are displayed below:

```
Customer Name: John Doe
Customer Email: john.doe@example.com
Statement processed.
```

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

Row level trigger

1. Create a row-level trigger that logs a change in an employee's salary. The trigger should only fire when the salary column is updated. It should insert a new record into an audit_log table that includes the old salary, the new salary, the date of the change, and the user who made the change.

DECLARE

```
v_customer_rec customer%ROWTYPE;
BEGIN
  SELECT *
    INTO v_customer_rec
   FROM customer
```

```

WHERE customer_id = 1;
DBMS_OUTPUT.PUT_LINE('Customer Name: ' || v_customer_rec.first_name || ' ' ||
v_customer_rec.last_name);
DBMS_OUTPUT.PUT_LINE('Customer Email: ' || v_customer_rec.email);
END;
/

```

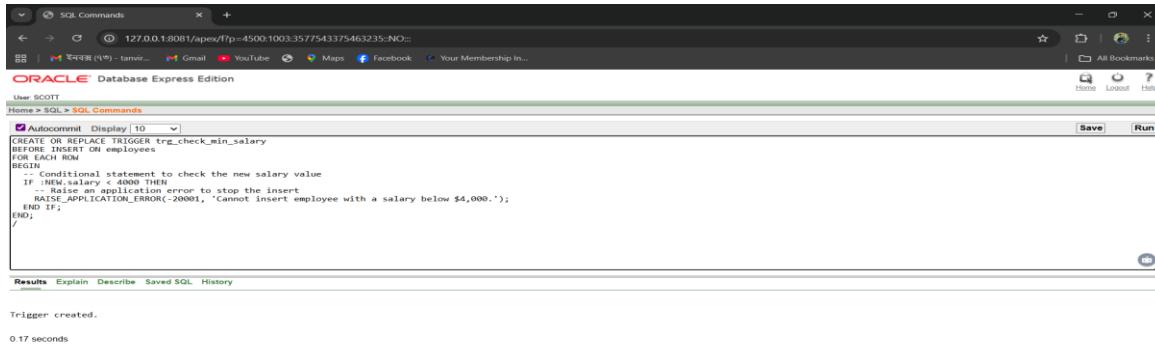
The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a PL/SQL block is run. The code declares a cursor for customer ID 101, fetches the first row into a record variable v_customer_rec, and then prints the customer's name and email using DBMS_OUTPUT.PUT_LINE. The results show the output: Customer Name: John Doe and Customer Email: john.doe@example.com.

2. Create a row-level trigger named trg_check_min_salary that prevents the insertion of a new employee if their salary is below \$4,000.

```

CREATE OR REPLACE TRIGGER trg_check_min_salary
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
  IF :NEW.salary < 4000 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Cannot insert employee with a salary below
$4,000.');
  END IF;
END;
/

```



User SCOTT

Home > SQL > SQL Commands

```
CREATE OR REPLACE TRIGGER trg_check_min_salary
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
    --Conditional statement to check the new salary value
    IF :NEW.salary < 4000 THEN
        --Raise an application error to stop the insert
        RAISE_APPLICATION_ERROR(-20001, 'Cannot insert employee with a salary below $4,000.');
    END IF;
END;
```

Results Explain Describe Saved SQL History

Trigger created.

0.17 seconds



Statement level trigger

1. Create a statement-level trigger that prevents any UPDATE operation on the employees table outside of standard business hours (8 AM to 5 PM).

CREATE OR REPLACE TRIGGER trg_prevent_off_hours_update

BEFORE UPDATE ON employees

BEGIN

IF TO_CHAR(SYSDATE, 'HH24') NOT BETWEEN '08' AND '17' THEN

RAISE_APPLICATION_ERROR(-20002, 'Updates are only allowed between 8 AM and 5 PM.');

END IF;

END;

/



User SCOTT

Home > SQL > SQL Commands

```
CREATE OR REPLACE TRIGGER trg_prevent_off_hours_update
BEFORE UPDATE ON employees
BEGIN
    IF TO_CHAR(SYSDATE, 'HH24') NOT BETWEEN '08' AND '17' THEN
        RAISE_APPLICATION_ERROR(-20002, 'Updates are only allowed between 8 AM and 5 PM.');
    END IF;
END;
```

Results Explain Describe Saved SQL History

Trigger created.

0.05 seconds



2. Modify the statement-level trigger to prevent any UPDATE operation on the employees table on weekends.

```
CREATE OR REPLACE TRIGGER trg_prevent_weekend_updates
BEFORE UPDATE ON employees
BEGIN
    IF TO_CHAR(SYSDATE, 'DY') IN ('SAT', 'SUN') THEN
        RAISE_APPLICATION_ERROR(-20001, 'Updates are not allowed on weekends.');
    END IF;
END;
/
```

The screenshot shows a browser window for Oracle Database Express Edition. The URL is 127.0.0.1:8081/apex/f?p=4500:1003:3577543375463235::NO:::. The page displays the SQL command for creating the trigger:

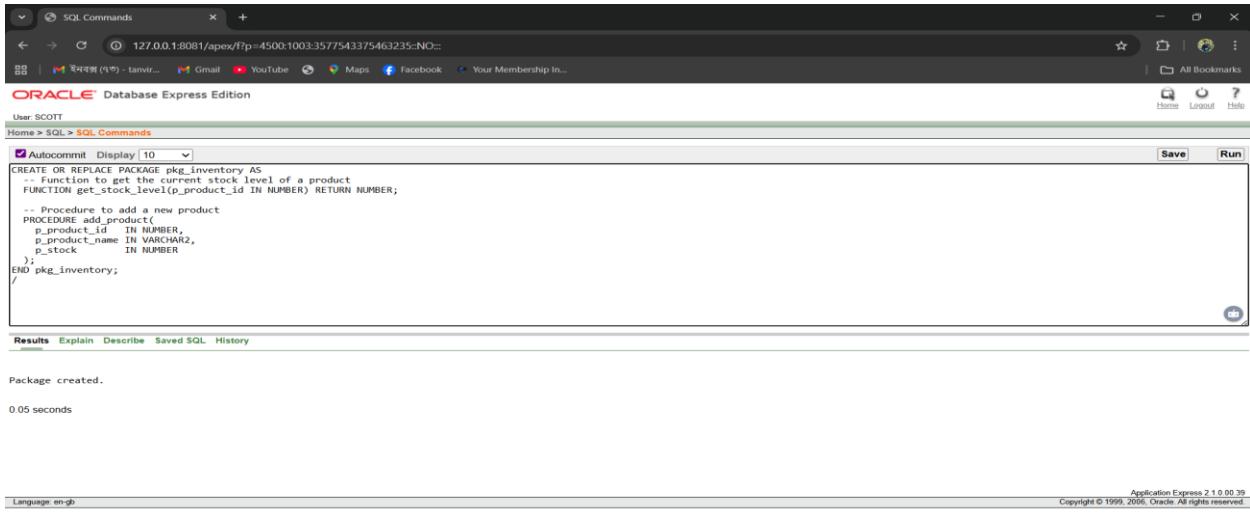
```
CREATE OR REPLACE TRIGGER trg_prevent_weekend_updates
BEFORE UPDATE ON employees
BEGIN
    -- Check if the current day is a weekend
    IF TO_CHAR(SYSDATE, 'DY') IN ('SAT', 'SUN') THEN
        RAISE_APPLICATION_ERROR(-20001, 'Updates are not allowed on weekends.');
    END IF;
END;
/
```

Below the code, it says "Trigger created." and "0.03 seconds". At the bottom right, it shows "Application Express 2.1.0.0.39" and the date "9/17/2025".

Package

1.Create a package named pkg_inventory that contains a function to check a product's stock level and a procedure to add new products.

```
CREATE OR REPLACE PACKAGE pkg_inventory AS
    FUNCTION get_stock_level(p_product_id IN NUMBER) RETURN NUMBER;
    PROCEDURE add_product(
        p_product_id  IN NUMBER,
        p_product_name IN VARCHAR2,
        p_stock      IN NUMBER
    );
END pkg_inventory;
/
```



The screenshot shows the Oracle Database Express Edition SQL Commands interface. A package named `pkg_inventory` is being created. The code includes a function `get_stock_level` and a procedure `add_product`. The output shows the package was created successfully in 0.05 seconds.

```

CREATE OR REPLACE PACKAGE pkg_inventory AS
    -- Function to get the current stock level of a product
    FUNCTION get_stock_level(p_product_id IN NUMBER) RETURN NUMBER;
    -- Procedure to add a new product
    PROCEDURE add_product(
        p_product_id IN NUMBER,
        p_product_name IN VARCHAR2,
        p_stock IN NUMBER
    );
END pkg_inventory;
/

```

Package created.
0.05 seconds

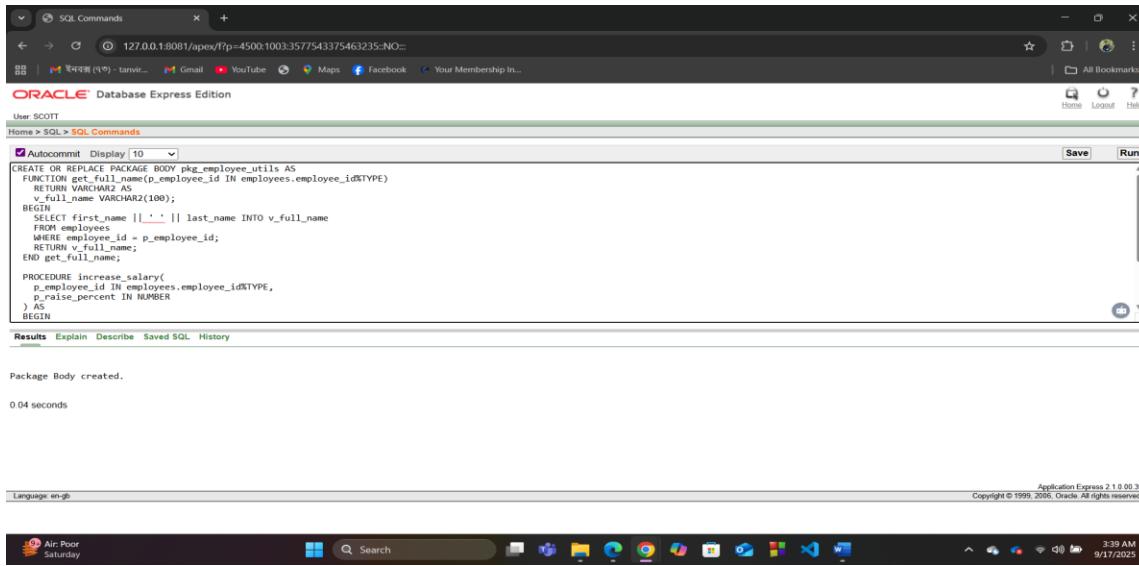
Package Body:

```

CREATE OR REPLACE PACKAGE BODY pkg_employee_utils AS
    FUNCTION get_full_name(p_employee_id IN employees.employee_id%TYPE)
        RETURN VARCHAR2 AS
            v_full_name VARCHAR2(100);
    BEGIN
        SELECT first_name || ' ' || last_name INTO v_full_name
        FROM employees
        WHERE employee_id = p_employee_id;
        RETURN v_full_name;
    END get_full_name;

    PROCEDURE increase_salary(
        p_employee_id IN employees.employee_id%TYPE,
        p_raise_percent IN NUMBER
    ) AS
    BEGIN
        UPDATE employees
        SET salary = salary * (1 + p_raise_percent / 100)
        WHERE employee_id = p_employee_id;
        COMMIT;
    END increase_salary;
END pkg_employee_utils;
/

```



The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, a package named pkg_employee_utils is being created. The code includes a function get_full_name that concatenates first and last names from the employees table, and a procedure increase_salary that raises the salary of an employee by a given percentage. The results pane shows the message "Package Body created." and a timestamp of "0.04 seconds". The status bar at the bottom indicates "Language: en-gb" and "Application Express 2.1.0.0.39 Copyright © 1999, 2005, Oracle. All rights reserved."

```

CREATE OR REPLACE PACKAGE pkg_employee_utils AS
  FUNCTION get_full_name(p_employee_id IN employees.employee_id%TYPE)
    RETURN VARCHAR2 AS
    v_full_name VARCHAR2(100);
  BEGIN
    SELECT first_name || ' ' || last_name INTO v_full_name
    FROM employees
    WHERE employees.id = p_employee_id;
    RETURN v_full_name;
  END get_full_name;

  PROCEDURE increase_salary(
    p_employee_id IN employees.employee_id%TYPE,
    p_raise_percent IN NUMBER
  );
  BEGIN
    ...
  END;
END;
  
```

2. Create a package to manage inventory. It should include a procedure to add a new product and a function to check the current stock level. This package should use a private function to validate the stock level before adding a product

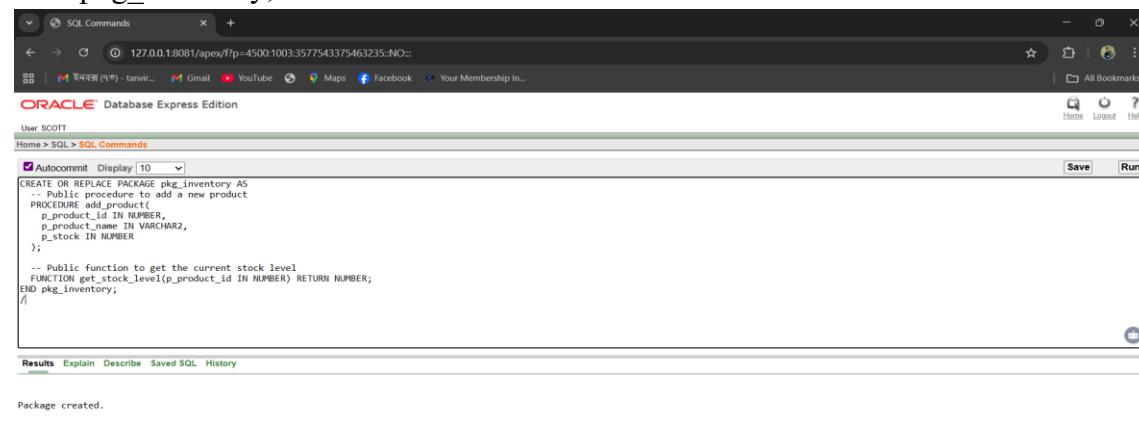
CREATE OR REPLACE PACKAGE pkg_inventory AS

```

  PROCEDURE add_product(
    p_product_id IN NUMBER,
    p_product_name IN VARCHAR2,
    p_stock IN NUMBER
  );
  
```

```
  FUNCTION get_stock_level(p_product_id IN NUMBER) RETURN NUMBER;
```

```
END pkg_inventory;
```



The screenshot shows the Oracle Database Express Edition interface. A package named pkg_inventory is being created. It contains a private procedure add_product that adds a new product with its name and stock level, and a public function get_stock_level that returns the stock level for a given product ID. The results pane shows the message "Package created." and a timestamp of "0.03 seconds". The status bar at the bottom indicates "Language: en-gb" and "Application Express 2.1.0.0.39 Copyright © 1999, 2005, Oracle. All rights reserved."

```

CREATE OR REPLACE PACKAGE pkg_inventory AS
  -- Public procedure to add a new product
  PROCEDURE add_product(
    p_product_id IN NUMBER,
    p_product_name IN VARCHAR2,
    p_stock IN NUMBER
  );
  -- Public function to get the current stock level
  FUNCTION get_stock_level(p_product_id IN NUMBER) RETURN NUMBER;
END pkg_inventory;
  
```



Package Body:

```

CREATE OR REPLACE PACKAGE BODY pkg_employee_utils AS
  FUNCTION get_full_name(p_employee_id IN employees.employee_id%TYPE)
    RETURN VARCHAR2 AS
      v_full_name VARCHAR2(100);
  BEGIN
    SELECT first_name || ' ' || last_name INTO v_full_name
    FROM employees
    WHERE employee_id = p_employee_id;
    RETURN v_full_name;
  END get_full_name;
  PROCEDURE increase_salary(
    p_employee_id IN employees.employee_id%TYPE,
    p_raise_percent IN NUMBER
  ) AS
  BEGIN
    UPDATE employees
    SET salary = salary * (1 + p_raise_percent / 100)
    WHERE employee_id = p_employee_id;
    COMMIT;
  END increase_salary;
END pkg_employee_utils;
/

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL editor contains the package body code provided above. The code defines a package body named pkg_employee_utils with two functions: get_full_name and increase_salary. The get_full_name function concatenates first and last names from the employees table. The increase_salary procedure updates the salary of an employee by a specified percentage. The code ends with a slash (/) to indicate the end of the package body.

Package Body created.

0.00 seconds



Query Writing

Exceptions handling: (Basic PL/SQL)

Variable:

1. Handle situations where a SELECT...INTO statement in PL/SQL returns zero rows or more than one row.

DECLARE

```
v_employee_id employees.employee_id%TYPE := 999; -- Employee 999 doesn't exist
v_first_name employees.first_name%TYPE;
v_salary     employees.salary%TYPE;
```

BEGIN

```
    SELECT first_name, salary
    INTO v_first_name, v_salary
    FROM employees
    WHERE employee_id = v_employee_id;
```

DBMS_OUTPUT.PUT_LINE('Employee Found: ' || v_first_name);

EXCEPTION

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.PUT_LINE('Error: No employee found with ID ' || v_employee_id || '.');

WHEN TOO_MANY_ROWS THEN

DBMS_OUTPUT.PUT_LINE('Error: Multiple employees found with ID ' || v_employee_id || '.');

WHEN OTHERS THEN

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

END;

/

```
SQL Commands
127.0.0.1:8081/apex/f?p=4500:1003:2446822171285035::NO::
User SCOTT
Home > SQL > SQL Commands
DECLARE
    v_employee_id employees.employee_id%TYPE := 999; -- Employee 999 doesn't exist
    v_first_name employees.first_name%TYPE;
    v_salary     employees.salary%TYPE;
BEGIN
    SELECT first_name, salary
    INTO v_first_name, v_salary
    FROM employees
    WHERE employee_id = v_employee_id;
    DBMS_OUTPUT.PUT_LINE('Employee Found: ' || v_first_name);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error: No employee found with ID ' || v_employee_id || '.');
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Error: Multiple employees found with ID ' || v_employee_id || '.');
END;
/
Results Explain Describe Saved SQL History
Error: No employee found with ID 999.
Statement processed.
0.03 seconds
Application Express 2.1.0.0.39
Copyright © 1999, 2006, Oracle. All rights reserved.
Language: en-gb
8:01 PM
9/16/2023
```

2. when this PL/SQL block is executed, and which exception handler will catch it.

DECLARE

```

v_num1    NUMBER(5) := 99999;
v_num2    NUMBER(2) := 2;
v_result  NUMBER(5);

BEGIN
  v_result := v_num1 * v_num2;
  DBMS_OUTPUT.PUT_LINE('Result: ' || v_result);
EXCEPTION
  WHEN VALUE_ERROR THEN
    DBMS_OUTPUT.PUT_LINE('Error: The result exceeded the size of the v_result variable.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/

```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands window, the code is entered. When run, an error message is displayed in the Results pane: "Error: The result exceeded the size of the v_result variable." The status bar at the bottom indicates the statement was processed in 0.02 seconds.

Operators

1.prevent a PL/SQL block from failing if a division-by-zero operation occurs.

DECLARE

```

v_numerator NUMBER := 100;
v_denominator NUMBER := 0;
v_result    NUMBER;

BEGIN
  v_result := v_numerator / v_denominator;
  DBMS_OUTPUT.PUT_LINE('Result: ' || v_result);
EXCEPTION
  WHEN ZERO_DIVIDE THEN

```

```

DBMS_OUTPUT.PUT_LINE('Error: Division by zero is not allowed.');
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/

```

```

SQL Commands
127.0.0.1:8081/apex/f?p=4500:1003:2446822171285035::NO::
User SCOTT
Home > SQL > SQL Commands
AutoCommit Display | 10 | Save | Run |
DECLARE
    v_numerator NUMBER := 100;
    v_denominator NUMBER := 0;
    v_result NUMBER;
BEGIN
    v_result := v_numerator / v_denominator;
    DBMS_OUTPUT.PUT_LINE('Result: ' || v_result);
EXCEPTION
    WHEN ZERO_DIVIDE THEN
        DBMS_OUTPUT.PUT_LINE('Error: Division by zero is not allowed.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/
Results Explain Describe Saved SQL History
Error: Division by zero is not allowed.
Statement processed.
0.02 seconds
Language: en-gb
Application Express 2.1.0.00.39
Copyright © 1999, 2006, Oracle. All rights reserved.
8:14 PM 9/16/2025

```

2. Handle Logical Operator with Exception Handling

DECLARE

```

v_score1 NUMBER := 90;
v_score2 NUMBER := 85;
v_status VARCHAR2(20);

```

BEGIN

```
IF v_score1 >= 85 AND v_score2 >= 85 THEN
```

```
    v_status := 'Passed';
```

```
ELSE
```

```
    v_status := 'Failed';
```

```
END IF;
```

```
DBMS_OUTPUT.PUT_LINE('Status: ' || v_status);
```

EXCEPTION

WHEN OTHERS THEN

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

END;

/

```

SQL Commands
User SCOTT
Home > SQL > SQL Commands
Autocommit Display 10
v_score1 NUMBER := 85;
v_score2 VARCHAR2(20);
BEGIN
  IF v_score1 >= 85 AND v_score2 >= 85 THEN
    v_status := 'Passed';
  ELSE
    v_status := 'Failed';
  END IF;
  DBMS_OUTPUT.PUT_LINE('Status: ' || v_status);
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/
Results Explain Describe Saved SQL History
Status: Passed
Statement processed.
0.00 seconds

```

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

Single-row-function

1.Handle a data conversion failure when using functions like TO_DATE.

DECLARE

```
v_date_string VARCHAR2(20) := '20-Feb-XX';
v_date DATE;
```

BEGIN

```
v_date := TO_DATE(v_date_string, 'DD-MON-YYYY');
DBMS_OUTPUT.PUT_LINE('Converted Date: ' || v_date);
```

EXCEPTION

WHEN VALUE_ERROR THEN

```
DBMS_OUTPUT.PUT_LINE('Error: The date format is incorrect.');
```

WHEN OTHERS THEN

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

END;

/

```

SQL Commands
User SCOTT
Home > SQL > SQL Commands
Autocommit Display 10
DECLARE
  v_date_string VARCHAR2(20) := '20-Feb-XX';
  v_date DATE;
BEGIN
  v_date := TO_DATE(v_date_string, 'DD-MON-YYYY');
  DBMS_OUTPUT.PUT_LINE('Converted Date: ' || v_date);
EXCEPTION
  WHEN VALUE_ERROR THEN
    DBMS_OUTPUT.PUT_LINE('Error: The date format is incorrect.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/
Results Explain Describe Saved SQL History
An unexpected error occurred: ORA-01841: (full) year must be between -4713 and +9999, and not be 0
Statement processed.
0.00 seconds

```

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



2. Handle a data conversion failure when using functions like TO_NUMBER.

DECLARE

```
v_price_text VARCHAR2(10) := 'abc';
v_price      NUMBER;
BEGIN
    v_price := TO_NUMBER(v_price_text);
    DBMS_OUTPUT.PUT_LINE('Price: ' || v_price);
EXCEPTION
    WHEN VALUE_ERROR THEN
        DBMS_OUTPUT.PUT_LINE('Error: Could not convert "' || v_price_text || '" to a number.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
```

/

```
SQL Commands
User SCOTT
Home > SQL > SQL Commands
AutoCommit Display: 10 Save Run
DECLARE
    v_price_text VARCHAR2(10) := 'abc';
    v_price      NUMBER;
BEGIN
    v_price := TO_NUMBER(v_price_text);
    DBMS_OUTPUT.PUT_LINE('Price: ' || v_price);
EXCEPTION
    WHEN VALUE_ERROR THEN
        DBMS_OUTPUT.PUT_LINE('Error: Could not convert "' || v_price_text || '" to a number.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/
Results Explain Describe Saved SQL History
Error: Could not convert "abc" to a number.
Statement processed.
0.00 seconds
```

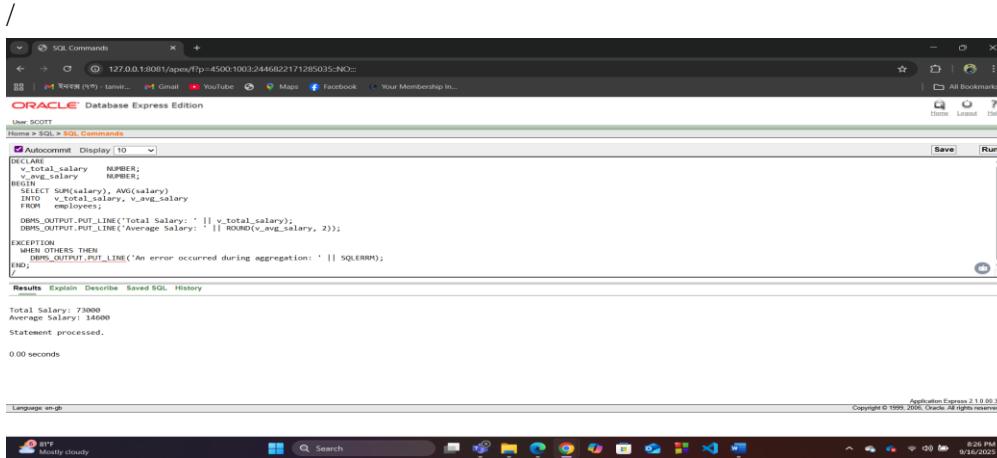


Group Function

1. Handle potential errors when aggregating data with functions like AVG.

DECLARE

```
v_total_salary  NUMBER;
v_avg_salary    NUMBER;
BEGIN
    SELECT SUM(salary), AVG(salary)
    INTO  v_total_salary, v_avg_salary
    FROM employees;
    DBMS_OUTPUT.PUT_LINE('Total Salary: ' || v_total_salary);
    DBMS_OUTPUT.PUT_LINE('Average Salary: ' || ROUND(v_avg_salary, 2));
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred during aggregation: ' || SQLERRM);
END;
```



```

DECLARE
  v_total_salary NUMBER;
  v_avg_salary NUMBER;
BEGIN
  SELECT SUM(salary), AVG(salary)
  INTO v_total_salary, v_avg_salary
  FROM employees;
  DBMS_OUTPUT.PUT_LINE('Total Salary: ' || v_total_salary);
  DBMS_OUTPUT.PUT_LINE('Average Salary: ' || ROUND(v_avg_salary, 2));
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred during aggregation: ' || SQLERRM);
END;
/

```

Results Explain Describe Saved SQL History

Total Salary: 73000
Average Salary: 14600
Statement processed.
0.00 seconds

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

2. Handle potential errors when aggregating data with functions like min,max.

DECLARE

```
v_max_salary    NUMBER;
v_min_salary    NUMBER;
```

BEGIN

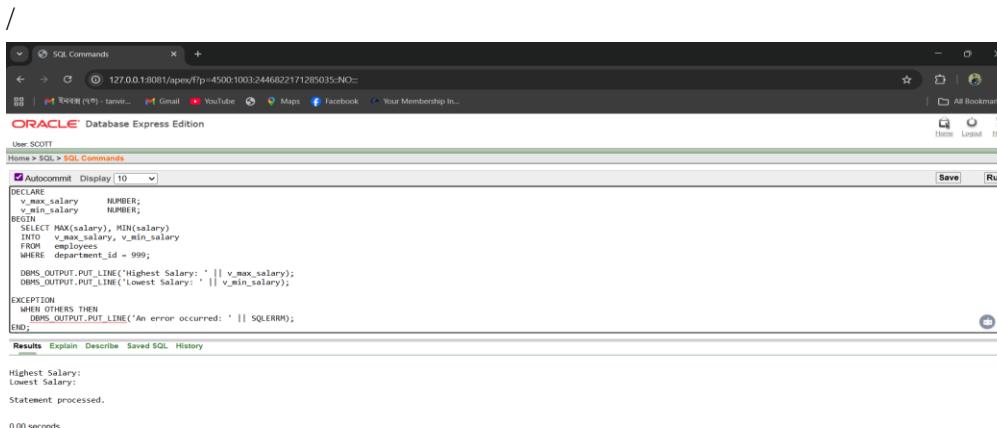
```
SELECT MAX(salary), MIN(salary)
INTO v_max_salary, v_min_salary
FROM employees
WHERE department_id = 999;
DBMS_OUTPUT.PUT_LINE('Highest Salary: ' || v_max_salary);
DBMS_OUTPUT.PUT_LINE('Lowest Salary: ' || v_min_salary);
```

EXCEPTION

WHEN OTHERS THEN

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

END;



```

/

```

```

DECLARE
  v_max_salary    NUMBER;
  v_min_salary    NUMBER;
BEGIN
  SELECT MAX(salary), MIN(salary)
  INTO v_max_salary, v_min_salary
  FROM employees
  WHERE department_id = 999;
  DBMS_OUTPUT.PUT_LINE('Highest Salary: ' || v_max_salary);
  DBMS_OUTPUT.PUT_LINE('Lowest Salary: ' || v_min_salary);
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/

```

Results Explain Describe Saved SQL History

Highest Salary:
Lowest Salary:
Statement processed.
0.00 seconds

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



Loop

1. Ensure a FOR loop handles an error in its processing logic without terminating the entire PL/SQL block.

BEGIN

FOR i IN 1..5 LOOP

BEGIN

-- Simulate a divide by zero error on the 3rd iteration

IF i = 3 THEN

DBMS_OUTPUT.PUT_LINE(10 / 0);

END IF;

DBMS_OUTPUT.PUT_LINE('Processed item ' || i);

EXCEPTION

WHEN ZERO_DIVIDE THEN

DBMS_OUTPUT.PUT_LINE('Skipping item ' || i || ' due to a zero divide error.');

END;

END LOOP;

DBMS_OUTPUT.PUT_LINE('Loop completed successfully.');

END;

/

```

User SCOTT
Home > SQL> SQL Commands
Autocommit: Display: 10 | Save | Run
BEGIN
FOR i IN 1..5 LOOP
BEGIN
-- Simulate a divide by zero error on the 3rd iteration
IF i = 3 THEN
DBMS_OUTPUT.PUT_LINE(10 / 0);
END IF;
DBMS_OUTPUT.PUT_LINE('Processed item ' || i);
EXCEPTION
WHEN ZERO_DIVIDE THEN
DBMS_OUTPUT.PUT_LINE('Skipping item ' || i || ' due to a zero divide error.');
END;
END LOOP;
DBMS_OUTPUT.PUT_LINE('Loop completed successfully.');
END;
/

```

Results Explain Describe Saved SQL History

Processed item 1
Processed item 2
Skipping item 3 due to a zero divide error.
Processed item 4
Processed item 5
Loop completed successfully.

Statement processed.

0.01 seconds

Language: en-gb Application Express 2.1 0.00.39 Copyright © 1995, 2006, Oracle. All rights reserved.

2. Ensure a While loop handles an error in its processing logic without terminating the entire PL/SQL block.

DECLARE

v_counter NUMBER := 1;

BEGIN

```

WHILE v_counter <= 5 LOOP
BEGIN
  IF v_counter = 4 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Invalid data encountered.');
  END IF;
  DBMS_OUTPUT.PUT_LINE('Processing step ' || v_counter);
  v_counter := v_counter + 1;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred. Terminating loop.');
    EXIT; -- Exit the loop on any error
END;
END LOOP;
END;
/

```

```

SQL Commands
127.0.0.1:8081/app/F?p=4500:1003:2446822171285035::NO::
ORACLE Database Express Edition
User SCOTT
Home > SQL > SQL Commands
Autocommit: Display / 10
Save Run
WHILE v_counter <= 5 LOOP
  BEGIN
    IF v_counter = 4 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Invalid data encountered.');
    END IF;
    DBMS_OUTPUT.PUT_LINE('Processing step ' || v_counter);
    v_counter := v_counter + 1;
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred. Terminating loop.');
      EXIT; -- Exit the loop on any error
  END;
END LOOP;
/

```

Results Explain Describe Saved SQL History

```

Processing step 1
Processing step 2
Processing step 3
An error occurred. Terminating loop.
Statement processed.

0.02 seconds

```

Language: en-gb Copyright © 1999, 2005, Oracle. All rights reserved.

Conditional Statements

1. An IF statement trigger an exception that needs to be handled, even if the logic itself is correct.

DECLARE

```

v_employee_id NUMBER := 999;
v_salary     NUMBER;
BEGIN
  IF v_employee_id IS NOT NULL THEN
    SELECT salary
    INTO v_salary
    FROM employees
    WHERE employee_id = v_employee_id;
  END IF;
END;
/

```

```

END IF;
DBMS_OUTPUT.PUT_LINE('Salary: ' || v_salary);
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Error: No employee found with the given ID.');
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/

```

```

User: SCOTT
Home > SQL > SQL Commands
Autocommit: Display: 10
IF v_employee_id IS NOT NULL THEN
  SELECT salary
  INTO v_salary
  FROM employees
  WHERE employee_id = v_employee_id;
END IF;

DBMS_OUTPUT.PUT_LINE('Salary: ' || v_salary);

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Error: No employee found with the given ID.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/

```

Results Explain Describe Saved SQL History

Error: No employee found with the given ID.
Statement processed.

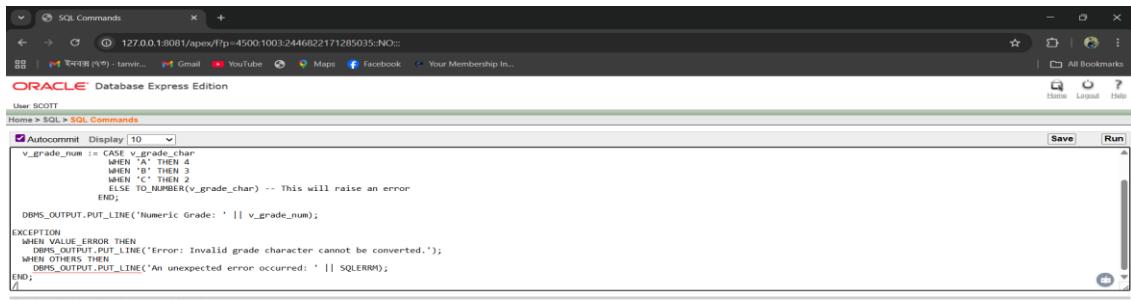
0.01 seconds

2. An CASE statement trigger an exception that needs to be handled, even if the logic itself is correct.

```

DECLARE
v_grade_char VARCHAR2(2) := 'X';
v_grade_num NUMBER;
BEGIN
v_grade_num := CASE v_grade_char
WHEN 'A' THEN 4
WHEN 'B' THEN 3
WHEN 'C' THEN 2
ELSE TO_NUMBER(v_grade_char) -- This will raise an error
END;
DBMS_OUTPUT.PUT_LINE('Numeric Grade: ' || v_grade_num);
EXCEPTION
WHEN VALUE_ERROR THEN
DBMS_OUTPUT.PUT_LINE('Error: Invalid grade character cannot be converted.');
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;

```



```

/
SQL Commands 127.0.1.8081/apex/f?p=4500:103:2446822171285035::NO::
ORACLE Database Express Edition
User SCOTT
Home > SQL > SQL Commands
Autocommit Display: 10
v_grade_num := CASE v_grade_char
    WHEN 'A' THEN 4
    WHEN 'B' THEN 3
    WHEN 'C' THEN 2
    ELSE TO_NUMBER(v_grade_char) -- This will raise an error
END;
DBMS_OUTPUT.PUT_LINE('Numeric Grade: ' || v_grade_num);

EXCEPTION
    WHEN VALUE_ERROR THEN
        DBMS_OUTPUT.PUT_LINE('Error: Invalid grade character cannot be converted.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/
Results Explain Describe Saved SQL History
Error: Invalid grade character cannot be converted.
Statement processed.
0.00 seconds
Language: en-US
Copyright © 1999, 2006, Oracle. All rights reserved.
Application Express 2.1.0.0.39
8:33 PM 9/16/2025

```

Subqueries

1. Handle when a SELECT...INTO statement uses a scalar subquery.

DECLARE

```
v_employee_name VARCHAR2(100);
v_dept_id      employees.department_id%TYPE;
```

BEGIN

```
SELECT first_name || ' ' || last_name, department_id
INTO  v_employee_name, v_dept_id
FROM  employees
```

WHERE salary = (SELECT MAX(salary) FROM employees); -- This subquery should return one row

```
DBMS_OUTPUT.PUT_LINE('Highest Earner: ' || v_employee_name);
```

EXCEPTION

WHEN TOO_MANY_ROWS THEN

```
DBMS_OUTPUT.PUT_LINE('Error: More than one employee has the highest salary.');
```

WHEN NO_DATA_FOUND THEN

```
DBMS_OUTPUT.PUT_LINE('Error: No data found.');
```

WHEN OTHERS THEN

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

END;

/

The screenshot shows a browser window for Oracle Database Express Edition. The URL is 127.0.0.1:8081/apex/f?p=4500:1003:2446822171285035::NO:::. The page displays a SQL command window with the following code:

```

SELECT first_name || ' ' || last_name, department_id
INTO v_employee_name, v_dept_id
FROM employees
WHERE salary = (SELECT MAX(salary) FROM employees); -- This subquery should return one row
DBMS_OUTPUT.PUT_LINE('Highest Earner: ' || v_employee_name);

EXCEPTION
WHEN TOO_MANY_ROWS THEN
DBMS_OUTPUT.PUT_LINE('Error: More than one employee has the highest salary.');
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Error: No data found.');
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/

```

The results show "Highest Earner: Steven King". Below the results, it says "Statement processed." and "0.00 seconds". At the bottom, it shows "Language: en-gb" and "Application Express 2.1.0.00.39 Copyright © 1999, 2005, Oracle. All rights reserved." The Windows taskbar at the bottom shows various icons and the date/time as 9/16/2025.

2. Handle when a SELECT...INTO statement uses a scalar subquery.

DECLARE

```

v_employee_name VARCHAR2(100);
v_dept_id      employees.department_id%TYPE;
BEGIN

```

```

    SELECT first_name || ' ' || last_name, department_id
    INTO v_employee_name, v_dept_id
    FROM employees
    WHERE salary = (SELECT MIN(salary) FROM employees); -- Changed MAX to MIN
    DBMS_OUTPUT.PUT_LINE('Lowest Earner: ' || v_employee_name);
EXCEPTION

```

WHEN TOO_MANY_ROWS THEN

DBMS_OUTPUT.PUT_LINE('Error: More than one employee has the lowest salary.');

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.PUT_LINE('Error: No data found.');

WHEN OTHERS THEN

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

END;/

```

SQL Commands
127.0.0.1:8081/apex/f?p=4500:1003:2446622171285035::NO=
User SCOTT
Home > SQL > SQL Commands
Autocommit Display: 10 Save Run
DECLARE
  v_employee_name VARCHAR2(100);
  v_dept_id NUMBER;
  v_dept_name VARCHAR2(100);
  v_lowest_salary NUMBER;
BEGIN
  SELECT first_name || ' ' || last_name, department_id
  INTO v_employee_name, v_dept_id
  FROM employees
  WHERE salary = (SELECT MIN(salary) FROM employees); -- Changed MAX to MIN
  DBMS_OUTPUT.PUT_LINE('Lowest Earner: ' || v_employee_name); -- Changed the output message
EXCEPTION
  WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE('Error: More than one employee has the lowest salary.');
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Error: No data found.');
END;
/

```

Results Explain Describe Saved SQL History

Lowest Earner: Bruce Ernst
Statement processed.

0.00 seconds



Joining

1. Use a JOIN to find a product's name and its supplier's name, and handle to prevent the code from failing.

DECLARE

```

v_product_id  NUMBER := 102; -- Change this ID to test the code
v_product_name Products.product_name%TYPE;
v_supplier_name Suppliers.supplier_name%TYPE;
BEGIN
  SELECT P.product_name, S.supplier_name
  INTO v_product_name, v_supplier_name
  FROM Products P
  JOIN Suppliers S ON P.supplier_id = S.supplier_id
  WHERE P.product_id = v_product_id;
  DBMS_OUTPUT.PUT_LINE('Product: ' || v_product_name);
  DBMS_OUTPUT.PUT_LINE('Supplier: ' || v_supplier_name);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Error: No product found with ID ' || v_product_id);
  WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE('Error: Multiple products found with ID ' || v_product_id);
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/

```

```

SQL Commands
127.0.0.1:8881/spxv/Ftp-4500:1003-2446822171285035:NO:=
ORACLE Database Express Edition
User: SCOTT
Home > SQL > SQL Commands
Autocommit: Display: 10 Save Run
SELECT v_product_name, v.supplier_id = S.supplier_id
WHERE P.product_id = v.supplier_id;
DBMS_OUTPUT.PUT_LINE('Product: ' || v_product_name);
DBMS_OUTPUT.PUT_LINE('Supplier: ' || v.supplier_name);

EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Error: No product found with ID ' || v_product_id);
WHEN TOO_MANY_ROWS THEN
DBMS_OUTPUT.PUT_LINE('Error: Multiple products found with ID ' || v_product_id);
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/

```

Results Explain Describe Saved SQL History

Product: Notebook
Supplier: Office Supply Co.
Statement processed.
0.02 seconds

Language: en-gb Application Express 21.1.0.00.39 Copyright © 1995-2025, Oracle. All rights reserved.

2. Use a LEFT JOIN to find out a specific supplier provides, including suppliers who have no product.

DECLARE

```

v_supplier_id NUMBER := 1; -- Change to a supplier ID with no products to see the result
v_supplier_name Suppliers.supplier_name%TYPE;
v_product_name Products.product_name%TYPE;
BEGIN
SELECT S.supplier_name, P.product_name
INTO v_supplier_name, v_product_name
FROM Suppliers S
LEFT JOIN Products P ON S.supplier_id = P.supplier_id
WHERE S.supplier_id = v_supplier_id;
IF v_product_name IS NULL THEN
DBMS_OUTPUT.PUT_LINE('Supplier "' || v_supplier_name || " has no products listed.");
ELSE
DBMS_OUTPUT.PUT_LINE('Supplier "' || v_supplier_name || " provides product: ' || v_product_name);
END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Error: No supplier found with ID ' || v_supplier_id);
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/

```

```

SQL Commands
127.0.0.1:8081/apex/F?p=4500:1003:2446822171285035::NO::
User SCOTT
Home > SQL > SQL Commands
AutoCommit: Display: 10 Save Run
-- Check if a product was found (it will be NULL if not)
IF v_product_name IS NULL THEN
  DBMS_OUTPUT.PUT_LINE('Supplier "' || v_supplier_name || '" has no products listed.');
ELSE
  DBMS_OUTPUT.PUT_LINE('Supplier "' || v_supplier_name || " provides product: '" || v_product_name);
END IF;
/
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Error: No supplier found with ID ' || v_supplier_id);
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/
Results Explain Describe Saved SQL History
Supplier "Tech Goods Inc." provides product: Wireless Mouse
Statement processed.
0.05 seconds

```

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

Advance PL/SQL Exception handling

Store Function

1. Create a stored function that calculates an employee's annual salary and includes exception handling for cases where the employee does not exist.

```

CREATE OR REPLACE FUNCTION get_annual_salary (
  p_employee_id IN employees.employee_id%TYPE
) RETURN NUMBER IS
  v_annual_salary NUMBER;
  v_salary      employees.salary%TYPE;
  v_commission   employees.commission_pct%TYPE;
BEGIN
  SELECT salary, commission_pct
  INTO v_salary, v_commission
  FROM employees
  WHERE employee_id = p_employee_id;
  v_annual_salary := (v_salary * 12) + (NVL(v_commission, 0) * v_salary * 12);
  RETURN v_annual_salary;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Error: Employee ' || p_employee_id || ' not found.');
    RETURN NULL;
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
    RETURN NULL;
END;
/

```

```

CREATE OR REPLACE FUNCTION get_annual_salary (
    p_employee_id IN employees.employee_id%TYPE
) RETURN NUMBER IS
    v_annual_salary NUMBER;
    v_salary      employees.salary%TYPE;
    v_commission   employees.commission_pct%TYPE;
BEGIN
    SELECT salary, commission_pct
    INTO v_salary, v_commission
    FROM employees
    WHERE employee_id = p_employee_id;
    v_annual_salary := (v_salary * 12) + (NVL(v_commission, 0) * v_salary * 12);
    RETURN v_annual_salary;
END;

```

Function created.
0.05 seconds

2. Create a stored function that calculates an employee's annual salary and includes exception handling for cases where the employee does not exist.

```

CREATE OR REPLACE FUNCTION get_annual_salary (
    p_employee_id IN employees.employee_id%TYPE
) RETURN NUMBER IS
    v_annual_salary NUMBER;
    v_salary      employees.salary%TYPE;
    v_commission   employees.commission_pct%TYPE;
BEGIN
    SELECT salary, commission_pct
    INTO v_salary, v_commission
    FROM employees
    WHERE employee_id = p_employee_id;
    v_annual_salary := (v_salary * 10) + (NVL(v_commission, 0) * v_salary * 12);
    RETURN v_annual_salary;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error: Employee ' || p_employee_id || ' not found.');
        RETURN NULL;
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
        RETURN NULL;
END;
/

```

```

SQL Commands
User SCOTT
Home > SQL > SQL Commands
Autocommit: Display: 10
CREATE OR REPLACE FUNCTION get_annual_salary (
    p_employee_id IN employees.employee_id%TYPE
) RETURN NUMBER IS
    v_annual_salary NUMBER;
    v_salary          employees.salary%TYPE;
    v_commission      employees.commission_pct%TYPE;
BEGIN
    SELECT salary, commission_pct
    INTO   v_salary, v_commission
    FROM   employees
    WHERE  employee_id = p_employee_id;
    v_annual_salary := (v_salary * 12) + (NVL(v_commission, 0) * v_salary * 12);
    RETURN v_annual_salary;
END;
/

```

Function created.
0.00 seconds

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

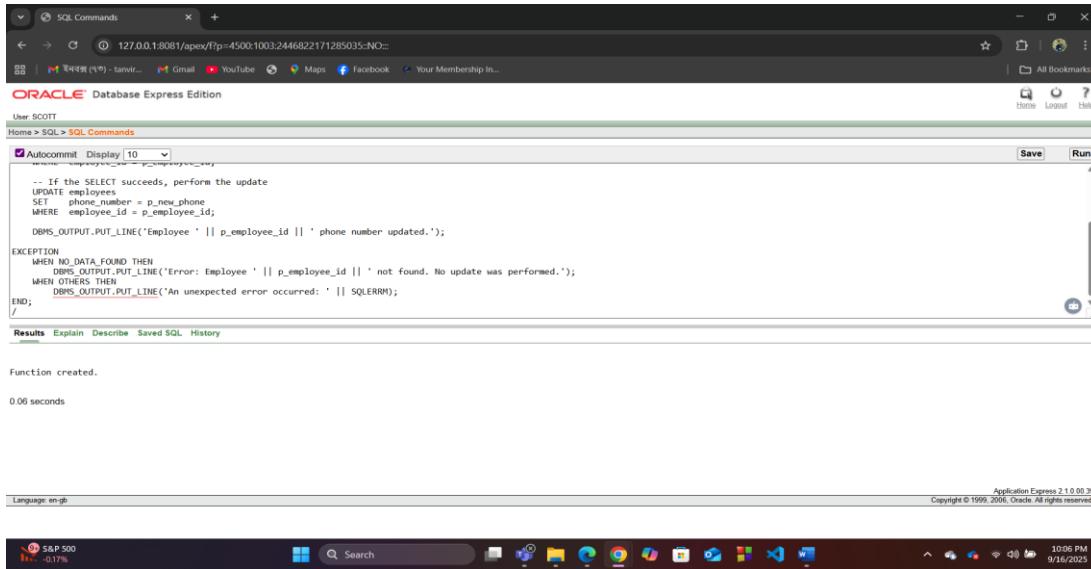
Stored Procedure

1. Create a stored procedure to update an employee's phone number and handle the case where the employee ID does not exist.

```

CREATE OR REPLACE PROCEDURE update_employee_phone (
    p_employee_id IN employees.employee_id%TYPE,
    p_new_phone   IN employees.phone_number%TYPE
) IS
    v_employee_id employees.employee_id%TYPE;
BEGIN
    SELECT employee_id INTO v_employee_id
    FROM   employees
    WHERE  employee_id = p_employee_id;
    UPDATE employees
    SET   phone_number = p_new_phone
    WHERE employee_id = p_employee_id;
    DBMS_OUTPUT.PUT_LINE('Employee ' || p_employee_id || ' phone number updated.');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error: Employee ' || p_employee_id || ' not found. No update was performed.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/

```



The screenshot shows the Oracle Database Express Edition SQL Commands interface. A stored procedure named 'update_employee_id' is being created. The code handles the case where the employee ID exists by updating the phone number, and handles the case where it does not exist by outputting an error message. The interface includes tabs for Results, Explain, Describe, Saved SQL, and History.

```

-- If the SELECT succeeds, perform the update
UPDATE employees
SET phone_number = p_new_phone
WHERE employee_id = p_employee_id;
DBMS_OUTPUT.PUT_LINE('Employee ' || p_employee_id || ' phone number updated.');

EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Error: Employee ' || p_employee_id || ' not found. No update was performed.');
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/

```

Function created.
0.00 seconds

Language: en-GB Application Express 2.1.0.00.39
Copyright © 1999, 2000, Oracle. All rights reserved.

2. Create a stored procedure to update an employee's phone number and handle the case where the employee ID does not exist.

CREATE OR REPLACE PROCEDURE insert_new_customer (

```

p_c_id    IN customer.c_id%TYPE,
p_c_name   IN customer.c_name%TYPE,
p_c_email  IN customer.c_email%TYPE
) IS
BEGIN

```

```

INSERT INTO customer (c_id, c_name, c_email)
VALUES (p_c_id, p_c_name, p_c_email);

```

DBMS_OUTPUT.PUT_LINE('New customer ' || p_c_name || ' inserted successfully.');

EXCEPTION

```

WHEN DUP_VAL_ON_INDEX THEN

```

DBMS_OUTPUT.PUT_LINE('Error: Customer with ID ' || p_c_id || ' already exists. No record was inserted.');

```

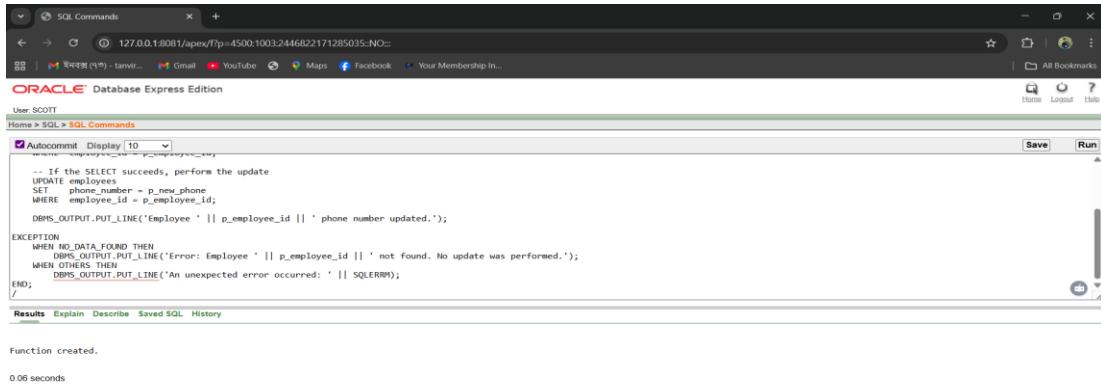
WHEN OTHERS THEN

```

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

END;

/



```

SQL Commands
127.0.0.1:8081/apex/f?p=4500:103:2446b2171285035--NO=
User SCOTT
Home > SQL > SQL Commands
AutoCommit Display: 10 Save Run
-- If the SELECT succeeds, perform the update
UPDATE employees
SET phone_number = p_new_phone
WHERE employee_id = p_employee_id;
DBMS_OUTPUT.PUT_LINE('Employee ' || p_employee_id || ' phone number updated.');

EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Error: Employee ' || p_employee_id || ' not found. No update was performed.');
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/
Results Explain Describe Saved SQL History
Function created.
0.06 seconds

```



Table-Based Record

1. Create a table-based record (%ROWTYPE) to fetch a single employee's data and handle exceptions if no record is found.

DECLARE

```
v_employee_id employees.employee_id%TYPE := 999; -- Change to an ID that exists to test
v_employee_rec employees%ROWTYPE;
```

BEGIN

SELECT *

```
INTO v_employee_rec
FROM employees
```

WHERE employee_id = v_employee_id;

```
DBMS_OUTPUT.PUT_LINE('Employee Found: ' || v_employee_rec.first_name || ' ' ||
v_employee_rec.last_name);
```

EXCEPTION

WHEN NO_DATA_FOUND THEN

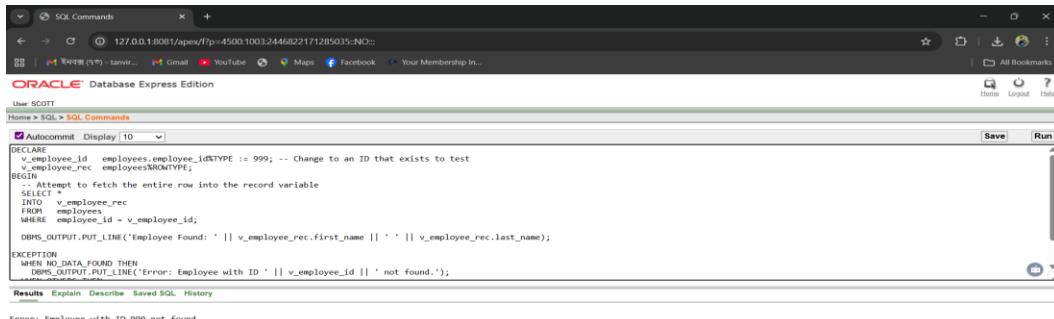
```
DBMS_OUTPUT.PUT_LINE('Error: Employee with ID ' || v_employee_id || ' not found.');
```

WHEN OTHERS THEN

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

END;

/



```

SQL Commands
User SCOTT
Home > SQL > SQL Commands
Autocommit Display | 10
DECLARE
    v_employee_id employees.employee_id%TYPE := 999; -- Change to an ID that exists to test
    v_employee_rec employees%ROWTYPE;
BEGIN
    -- Attempt to fetch the entire row into the record variable
    SELECT *
    INTO v_employee_rec
    FROM employees
    WHERE employee_id = v_employee_id;
    DBMS_OUTPUT.PUT_LINE('Employee Found: ' || v_employee_rec.first_name || ' ' || v_employee_rec.last_name);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error: Employee with ID ' || v_employee_id || ' not found.');
END;
/

```

Results Explain Describe Saved SQL History

Error: Employee with ID 999 not found.
Statement processed.

0.03 seconds

Language en_gb Application Express 2.0.0.39 Copyright © 1999-2006, Oracle. All rights reserved.

2.Find Multiple Records in a FOR loop.

BEGIN

FOR emp_rec IN (SELECT * FROM employees) LOOP

BEGIN

IF emp_rec.employee_id = 104 THEN

RAISE_APPLICATION_ERROR(-20001, 'Simulating an error for processing.');

END IF;

DBMS_OUTPUT.PUT_LINE('Processing employee: ' || emp_rec.first_name || ' ' || emp_rec.last_name);

EXCEPTION

WHEN OTHERS THEN

DBMS_OUTPUT.PUT_LINE('Warning: Skipping employee ' || emp_rec.employee_id || ' due to error: ' || SQLERRM);

END;

END LOOP;

DBMS_OUTPUT.PUT_LINE('Finished processing all employees.');

END;

/

```

SQL Commands
127.0.0.1:8081/oracle?port=4500:100:2446822171285035:NO:=
Home > SQL > SQL Commands
User SCOTT
AutoCommit Display: 10
Save Run
BEGIN
    -- The FOR loop implicitly handles NO_DATA_FOUND
    FOR emp_rec IN (SELECT * FROM employees) LOOP
        DBMS_OUTPUT.PUT_LINE('Processing employee: ' || emp_rec.first_name || ' ' || emp_rec.last_name);
        -- Simulate an error on a specific record (e.g., employee with ID 104)
        IF emp_rec.employee_id = 104 THEN
            RAISE_APPLICATION_ERROR(-20001, 'Simulating an error for processing.');
        END IF;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Warning: Skipping employee ' || emp_rec.employee_id || ' due to error: ' || SQLERRM);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Warning: Skipping employee ' || emp_rec.employee_id || ' due to error: ' || SQLERRM);
    END;
END;
/

```

Results Explain Describe Saved SQL History

Processing employee: Steven King
 Processing employee: Neena Kochhar
 Processing employee: Lex De Haan
 Processing employee: Bruce Ernster
 Processing employee: 104
 Error at line 10: ORA-20001: Simulating an error for processing.
 Processing employee: Bruce Ernster
 Finished processing all employees.

Statement processed.

0.03 seconds

Language: en_gb Application Express 21.0.0.0.39 Copyright © 1999-2025, Oracle. All rights reserved.

Explicit Cursor

1. Create an explicit cursor to fetch employee data and handle exceptions if no records are found.

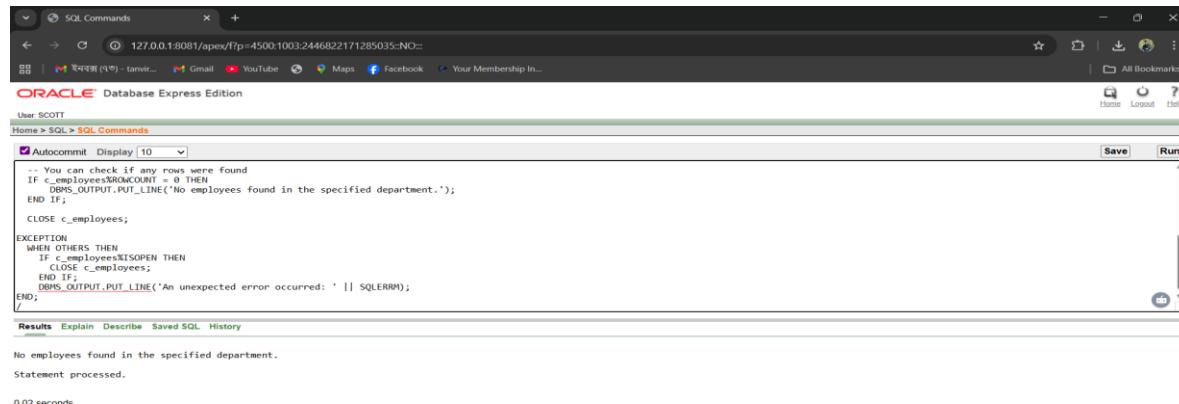
DECLARE

```

CURSOR c_employees IS
    SELECT first_name, last_name, salary
    FROM employees
    WHERE department_id = 999; -- Change to a valid department ID to test
    v_first_name employees.first_name%TYPE;
    v_last_name employees.last_name%TYPE;
    v_salary     employees.salary%TYPE;
BEGIN
    OPEN c_employees;
    LOOP
        FETCH c_employees INTO v_first_name, v_last_name, v_salary;
        EXIT WHEN c_employees%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(v_first_name || ' ' || v_last_name || ' has a salary of ' || v_salary);
    END LOOP;
    IF c_employees%ROWCOUNT = 0 THEN
        DBMS_OUTPUT.PUT_LINE('No employees found in the specified department.');
    END IF;
    CLOSE c_employees;
EXCEPTION
    WHEN OTHERS THEN
        IF c_employees%ISOPEN THEN
            CLOSE c_employees;
        END IF;

```

```
DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/
```



```
-- You can check if any rows were found
IF c_employees%ISOPEN = 0 THEN
  DBMS_OUTPUT.PUT_LINE('No employees found in the specified department.');
END IF;

CLOSE c_employees;

EXCEPTION
WHEN OTHERS THEN
  IF c_employees%ISOPEN THEN
    CLOSE c_employees;
  END IF;
  DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/
```

No employees found in the specified department.
Statement processed.
0.02 seconds

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

2. Create an explicit cursor to fetch employee data and handle exceptions too many rows?

DECLARE

```
CURSOR c_employee (p_salary IN employees.salary%TYPE) IS
  SELECT first_name, last_name
  FROM employees
  WHERE salary = p_salary;
  v_first_name employees.first_name%TYPE;
  v_last_name employees.last_name%TYPE;
BEGIN
  OPEN c_employee(24000);
  FETCH c_employee INTO v_first_name, v_last_name;
  IF c_employee%FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Employee found with salary 24000: ' || v_first_name || ' ' || v_last_name);
  END IF;
EXCEPTION
  WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE('Error: Multiple employees found with that salary.');
  IF c_employee%ISOPEN THEN
    CLOSE c_employee;
  END IF;
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
```

```

IF c_employee%ISOPEN THEN
  CLOSE c_employee;
END IF;
END;
/

```

```

SQL Commands
127.0.0.1:8081/apex/f?p=4500:1003:2446822171285035::NO::
ORACLE Database Express Edition
User SCOTT
Home > SQL > SQL Commands
Save Run
Autocommit Display / 10
EXCEPTION
-- This will be raised if the cursor fetches more than one row
WHEN TOO_MANY_ROWS THEN
  DBMS_OUTPUT.PUT_LINE('Error: Multiple employees found with that salary.');
  IF c_employee%ISOPEN THEN
    CLOSE c_employee;
  END IF;
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
    IF c_employee%ISOPEN THEN
      CLOSE c_employee;
    END IF;
  END IF;
Employee found with salary 24000: Steven King
Statement processed.
0.02 seconds

```



Cursor-Based Record

1.Create a cursor FOR loop to join two tables and handle potential errors.

```

DECLARE
  CURSOR c_customer_service IS
    SELECT c.customer_name, s.service_name
    FROM Customers c
    JOIN Services s ON c.customer_id = s.customer_id;
    v_customer_service_rec c_customer_service%ROWTYPE;
BEGIN
  OPEN c_customer_service;
  LOOP
    FETCH c_customer_service INTO v_customer_service_rec;
    EXIT WHEN c_customer_service%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Customer: ' || v_customer_service_rec.customer_name || ','
    Service: ' || v_customer_service_rec.service_name);
  END LOOP;
  CLOSE c_customer_service;
EXCEPTION
  WHEN OTHERS THEN
    IF c_customer_service%ISOPEN THEN
      CLOSE c_customer_service;
    END IF;

```

```

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

```

```

SQL Commands
127.0.0.1:8081/apex/?p=450010032446822171285035-NO
User SCOTT
Home > SQL > SQL Commands
Autocommit: Display: 10
-- The cursor is defined and opened implicitly by the FOR loop
FOR c.CUSTOMER_NAME IN (
  SELECT c.customer_name, s.service_name
  FROM Customers c
  JOIN Services s ON c.customer_id = s.customer_id
)
LOOP
  -- The record variable is automatically created and populated
  DBMS_OUTPUT.PUT_LINE('Customer: ' || cust_service_rec.customer_name || ', Service: ' || cust_service_rec.service_name);
END LOOP;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;

```

Results Explain Describe Saved SQL History

Customer: Alice, Service: Wash & Fold
Customer: Bob, Service: Dry Cleaning
Customer: Charlie, Service: Ironing

Statement processed.

0.07 seconds

Language: eng/zh
Copyright © 1999, 2005, Oracle. All rights reserved.
Application Express 2.1.0.0.39
11:41 PM 9/16/2025

2. Create an explicit cursor to join two tables and process the results one row at a time?

```

DECLARE
  CURSOR c_customer_service IS
    SELECT c.customer_name, s.service_name
    FROM Customers c
    JOIN Services s ON c.customer_id = s.customer_id;
    v_customer_service_rec c_customer_service%ROWTYPE;
BEGIN
  OPEN c_customer_service;
  LOOP
    FETCH c_customer_service INTO v_customer_service_rec;
    EXIT WHEN c_customer_service%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Customer: ' || v_customer_service_rec.customer_name || ',  

Service: ' || v_customer_service_rec.service_name);
  END LOOP;
  CLOSE c_customer_service;
EXCEPTION
  WHEN OTHERS THEN
    IF c_customer_service%ISOPEN THEN
      CLOSE c_customer_service;
    END IF;
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/

```

```

SQL Commands
127.0.0.1:8081/apex/f?p=4500:1003:2446822171285035:NQ=
ORACLE Database Express Edition
User SCOTT
Home > SQL > SQL Commands
Autocommit: On | Display: 10 | Save | Run
DBMS_OUTPUT.PUT_LINE('Customer: ' || v_customer_service_rec.customer_name || ', Service: ' || v_customer_service_rec.service_name);
END LOOP;
-- 5. Close the cursor to release resources
CLOSE c_customer_service;
EXCEPTION
  WHEN OTHERS THEN
    IF c_customer_service%ISOPEN THEN
      CLOSE c_customer_service;
    END IF;
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/
Results Explain Describe Saved SQL History
Customer: Alice, Service: Wash & Fold
Customer: Bob, Service: Dry Cleaning
Customer: Charlie, Service: Ironing
Statement processed.

0.07 seconds

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

```

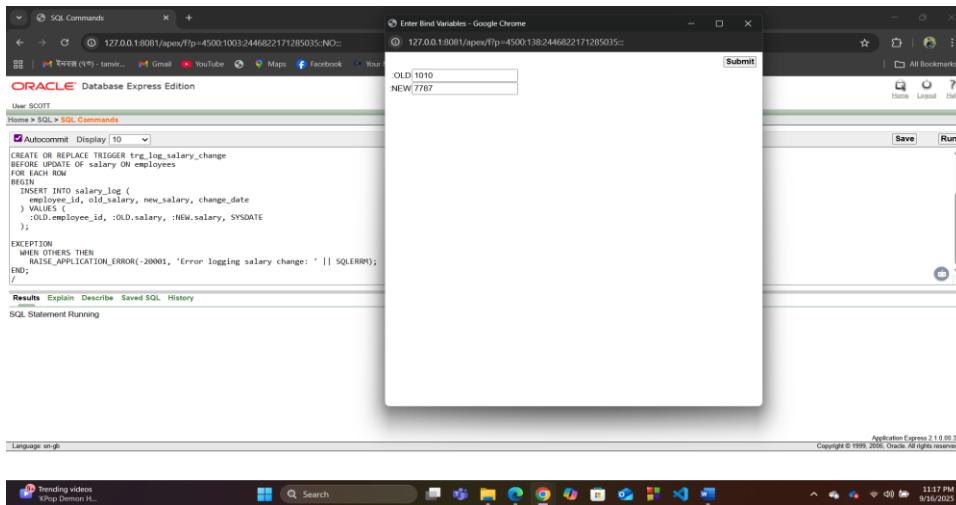
Row-level-trigger

1. Create a row-level trigger to automatically update a salary_log table whenever an employee's salary is changed, and what kind of exceptions should handle.

```

CREATE TABLE salary_log (
  employee_id NUMBER,
  old_salary NUMBER,
  new_salary NUMBER,
  change_date DATE
);
CREATE OR REPLACE TRIGGER trg_log_salary_change
BEFORE UPDATE OF salary ON employees
FOR EACH ROW
BEGIN
  INSERT INTO salary_log (
    employee_id, old_salary, new_salary, change_date
  ) VALUES (
    :OLD.employee_id, :OLD.salary, :NEW.salary, SYSDATE
  );
  EXCEPTION
    WHEN OTHERS THEN
      RAISE_APPLICATION_ERROR(-20001, 'Error logging salary change: ' || SQLERRM);
END;
/

```



2.Create a row-level trigger to automatically update a salary_log table whenever an employee's salary is changed, and what kind of exceptions should you handle.

CREATE OR REPLACE TRIGGER trg_validate_bill_due_date

BEFORE INSERT OR UPDATE ON Utilities_Bill

FOR EACH ROW

DECLARE

 e_invalid_due_date EXCEPTION;

BEGIN

 IF :NEW.u_due < SYSDATE THEN

 RAISE e_invalid_due_date;

 END IF;

EXCEPTION

 WHEN e_invalid_due_date THEN

 RAISE_APPLICATION_ERROR(-20002, 'The bill due date cannot be in the past.');

 WHEN OTHERS THEN

 RAISE_APPLICATION_ERROR(-20003, 'An unexpected error occurred in the trigger: ' || SQLERRM);

END;

/

```

SQL Commands
127.0.0.1:8081/oracle/p=4500:1003-24468221/1285035:NO::

User SCOTT
Home > SQL > SQL Commands
Autocommit Display 10 ▾
FOR EACH ROW
DECLARE
    e_invalid_due_date EXCEPTION;
BEGIN
    IF NEW.e_due < SYSDATE THEN
        RAISE e_invalid_due_date;
    END IF;
EXCEPTION
    WHEN e_invalid_due_date THEN
        RAISE_APPLICATION_ERROR(-20002, 'The bill due date cannot be in the past.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20003, 'An unexpected error occurred in the trigger: ' || SQLERRM);
END;
/

```

Results Explain Describe Saved SQL History

Trigger created.

0.08 seconds

Language: en-gb Copyright © 1995, 2006, Oracle. All rights reserved. Application Express 2.1.0.09.39

Statement level trigger

1. Create a statement-level trigger to prevent DELETE operations on a table outside of business hours, and what kind of exceptions should you handle.

CREATE OR REPLACE TRIGGER trg_prevent_weekend_delete

BEFORE DELETE ON employees

DECLARE

e_weekend_delete EXCEPTION;

BEGIN

IF TO_CHAR(SYSDATE, 'DY') IN ('SAT', 'SUN') THEN

RAISE e_weekend_delete;

END IF;

EXCEPTION

WHEN e_weekend_delete THEN

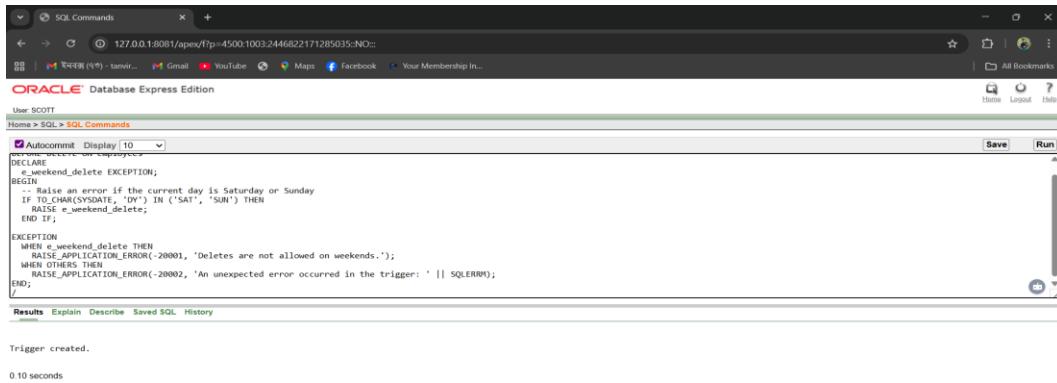
RAISE_APPLICATION_ERROR(-20001, 'Deletes are not allowed on weekends.');

WHEN OTHERS THEN

RAISE_APPLICATION_ERROR(-20002, 'An unexpected error occurred in the trigger: ' || SQLERRM);

END;

/



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

```

DECLARE
    e_weekend_delete EXCEPTION;
BEGIN
    -- Raise an error if the current day is Saturday or Sunday
    IF TO_CHAR(SYSDATE, 'DY') IN ('SAT', 'SUN') THEN
        RAISE e_weekend_delete;
    END IF;
EXCEPTION
    WHEN e_weekend_delete THEN
        RAISE_APPLICATION_ERROR(-20001, 'Deletes are not allowed on weekends.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'An unexpected error occurred in the trigger: ' || SQLERRM);
END;

```

Below the code, the message "Trigger created." is displayed, followed by "0.10 seconds". At the bottom right, it says "Application Express 2.1.0.00.39" and "Copyright © 1999-2006, Oracle. All rights reserved."



2. Create a statement-level trigger to prevent DELETE operations on a table outside of business hours, and what kind of exceptions should you handle.

CREATE OR REPLACE TRIGGER trg_restrict_dml_hours
BEFORE INSERT OR UPDATE OR DELETE ON Laundry_Service

DECLARE

 e_outside_hours EXCEPTION;

 v_current_hour NUMBER := TO_NUMBER(TO_CHAR(SYSDATE, 'HH24'));

BEGIN

 IF v_current_hour < 9 OR v_current_hour >= 17 THEN

 RAISE e_outside_hours;

 END IF;

EXCEPTION

 WHEN e_outside_hours THEN

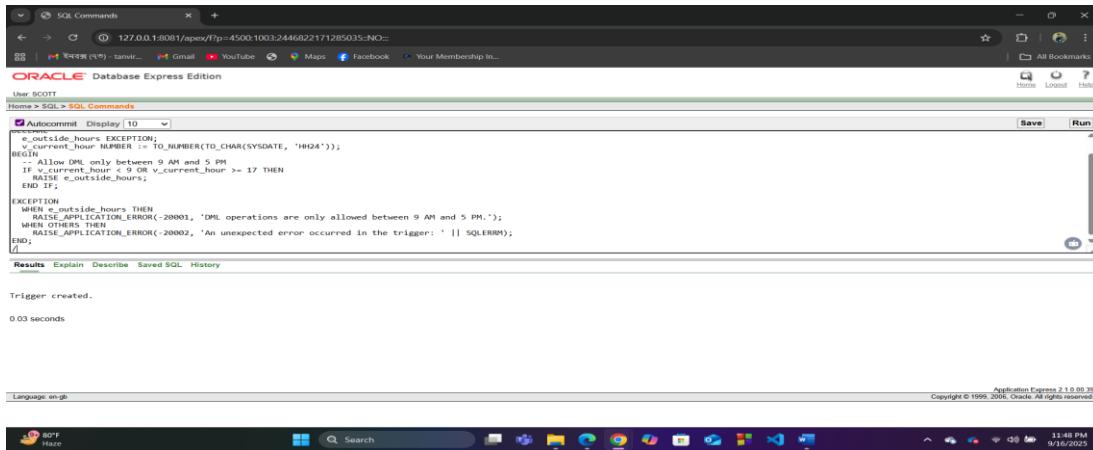
 RAISE_APPLICATION_ERROR(-20001, 'DML operations are only allowed between 9 AM and 5 PM.');

 WHEN OTHERS THEN

 RAISE_APPLICATION_ERROR(-20002, 'An unexpected error occurred in the trigger: ' || SQLERRM);

 END;

/



The screenshot shows the Oracle Database Express Edition SQL Commands interface. A trigger named 'outside_hours' is being created. The code checks if the current hour is between 9 AM and 5 PM. If it's outside this range, it raises an application error. The trigger is created successfully in 0.03 seconds.

```

SQL Commands
127.0.0.1:5001/spx/F=4500:1003:3446822171285035--NO...
ORACLE Database Express Edition
User: SCOTT
Home > SQL > SQL Commands
Autocommit: Display: 10 Save Run
--outside_hours EXCEPTION;
v_current_hour NUMBER := TO_NUMBER(TO_CHAR(SYSDATE, 'HH24'));
BEGIN
    -- Allow DML only between 9 AM and 5 PM
    IF v_current_hour < 9 OR v_current_hour >= 17 THEN
        RAISE outside_hours;
    END IF;
EXCEPTION
    WHEN outside_hours THEN
        RAISE_APPLICATION_ERROR(-20001, 'DML operations are only allowed between 9 AM and 5 PM.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'An unexpected error occurred in the trigger: ' || SQLERRM);
END;
/
Trigger created.
0.03 seconds
Language: en-gb Application Express 2.1.0.0.39
Copyright © 1999, 2006, Oracle. All rights reserved.
11:48 PM 9/16/2023

```

Package

1. Create a PL/SQL package to group related procedures and functions for managing Customers, and how can you centralize their exception handling.

CREATE OR REPLACE PACKAGE customer_pkg IS

```

FUNCTION get_customer_name(p_c_id IN customer.c_id%TYPE) RETURN VARCHAR2;
PROCEDURE update_customer_email(p_c_id IN customer.c_id%TYPE, p_new_email IN
customer.c_email%TYPE);
END customer_pkg;
/

```

CREATE OR REPLACE PACKAGE BODY customer_pkg IS

```

FUNCTION get_customer_name(p_c_id IN customer.c_id%TYPE) RETURN VARCHAR2 IS
    v_name customer.c_name%TYPE;

```

BEGIN

```

SELECT c_name
INTO v_name
FROM customer
WHERE c_id = p_c_id;
RETURN v_name;

```

EXCEPTION

WHEN NO_DATA_FOUND THEN

```

DBMS_OUTPUT.PUT_LINE('Error: Customer ' || p_c_id || ' not found.');
RETURN NULL;

```

WHEN OTHERS THEN

```

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

```

END get_customer_name;

PROCEDURE update_customer_email(p_c_id IN customer.c_id%TYPE, p_new_email IN
customer.c_email%TYPE) IS

BEGIN

```

UPDATE customer
SET  c_email = p_new_email
WHERE c_id = p_c_id;
IF SQL%ROWCOUNT = 0 THEN
    RAISE NO_DATA_FOUND;
END IF;
DBMS_OUTPUT.PUT_LINE('Customer ' || p_c_id || ' email updated.');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error: Customer ' || p_c_id || ' not found. Email not
updated.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END update_customer_email;
END customer_pkg;
/

```

The screenshot shows the Oracle Database Express Edition interface. The SQL Commands window contains the provided PL/SQL code. The output pane shows the results of the execution, including the update statement and an error message for a non-existent customer.

The screenshot shows the Windows taskbar at the bottom of the screen, indicating that the Oracle Database Express Edition application is currently active.

2. Get_customer_name function and update_customer_email procedure in a PL/SQL package using different programming constructs.

CREATE OR REPLACE PACKAGE customer_pkg IS

```

FUNCTION get_customer_name(p_c_id IN customer.c_id%TYPE) RETURN VARCHAR2;
PROCEDURE update_customer_email(p_c_id IN customer.c_id%TYPE, p_new_email IN
customer.c_email%TYPE);
END customer_pkg;
/

```

CREATE OR REPLACE PACKAGE BODY customer_pkg IS

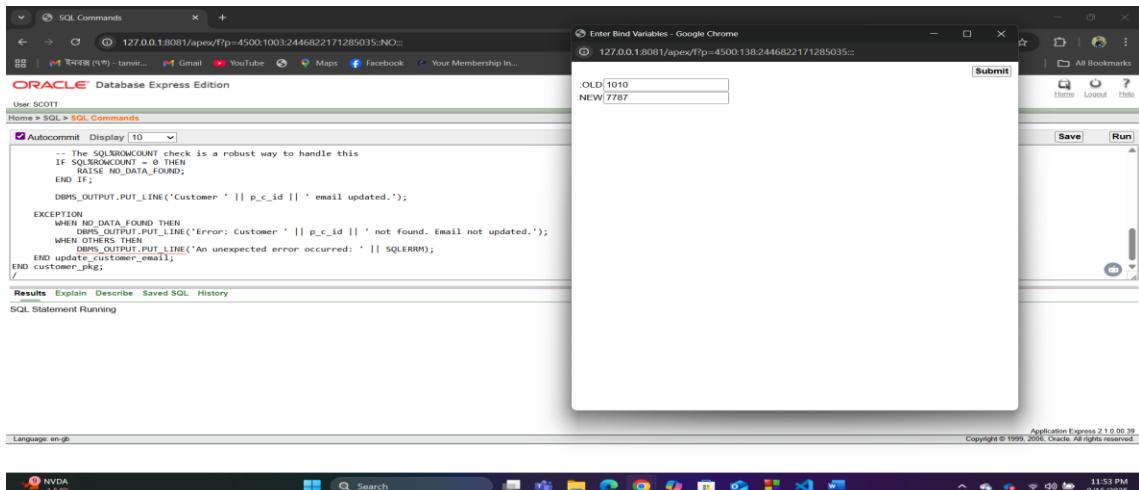
```
FUNCTION get_customer_name(p_c_id IN customer.c_id%TYPE) RETURN VARCHAR2 IS
```

```

v_name customer.c_name%TYPE := NULL;
BEGIN
  FOR rec IN (
    SELECT c_name
    FROM customer
    WHERE c_id = p_c_id
  ) LOOP
    v_name := rec.c_name;
  END LOOP;
  RETURN v_name;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
    RETURN NULL;
END get_customer_name;

PROCEDURE update_customer_email(p_c_id IN customer.c_id%TYPE, p_new_email IN
customer.c_email%TYPE) IS
BEGIN
  UPDATE customer
  SET c_email = p_new_email
  WHERE c_id = p_c_id;
  IF SQL%ROWCOUNT = 0 THEN
    RAISE NO_DATA_FOUND;
  END IF;
  DBMS_OUTPUT.PUT_LINE('Customer ' || p_c_id || ' email updated.');
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Error: Customer ' || p_c_id || ' not found. Email not
updated.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END update_customer_email;
END customer_pkg;
/

```



Relational Algebra:

Question 1: Find the first and last names of customers whose first name is "John".

Answer:

$$\prod_{first_name, last_name} (\sigma_{first_name = "John"} (Customer))$$

Question 2: Find the email addresses of all customers with a last name of "Smith".

Answer:

$$\prod_{email} (\sigma_{last_name = "Smith"} (Customer))$$

Question-3: Find the names of all laundry services that cost more than \$5.00.

Answer:

$$\prod_{service_name} (\sigma_{price > 5.00} (Laundry_Service))$$

Question-3: Find job titles all employees who have a salary greater than \$40,000.

Answer:

$$\prod_{job_title} (\sigma_{salary > 40000} (Employee))$$

Question 4: Find the details of employees whose first name is "David".

Answer:

$$\prod_{employee_id, first_name, last_name, job_title, hire_date, salary} (\sigma_{first_name = "David"} (Employee))$$

Question -5: Find the names and prices of all laundry services that cost less than or equal to \$2.50.

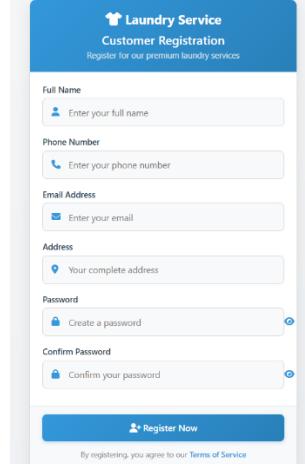
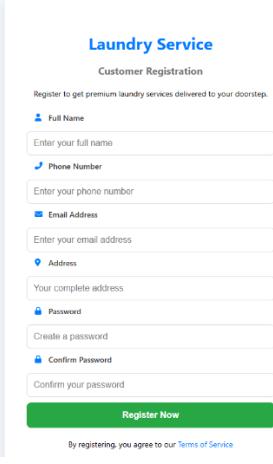
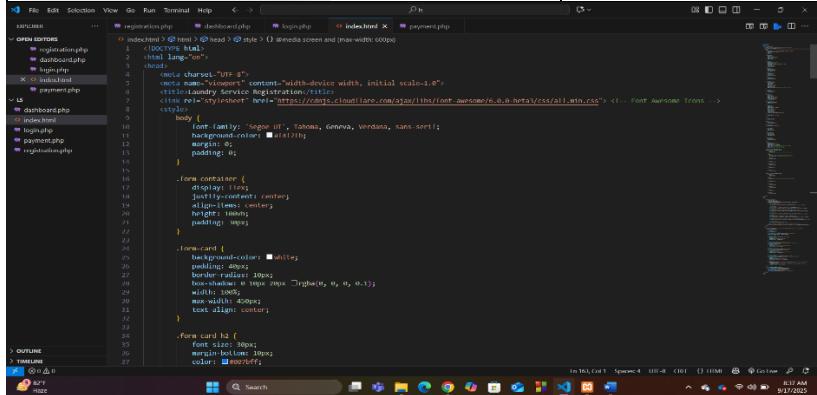
Answer:

$\Pi \text{service_name, price} (\sigma_{\text{price} \leq 2.50} (\text{Laundry_Service}))$

Bonus

1. User interface & code:

Registration:

Mid Interface	Final Interface
 	

Login:

Laundry Service

Welcome Back!

Login to access your account

Welcome to our laundry service! We're glad to have you back.

Phone
Email

Phone Number

Enter your phone number

Password

Enter your password

Remember me
[Forgot Password?](#)

Login

Don't have an account? [Register now](#)

Laundry Service

Welcome Back! Login to access your account

Phone / Email

Enter your email

Password

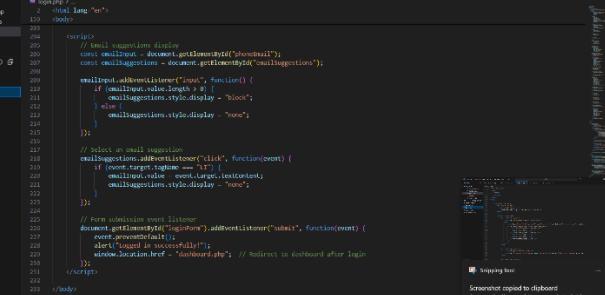
Enter your password

Remember me

Login

[Forgot your password?](#)

Back



The screenshot shows a browser's developer tools open, specifically the DOM tab. The page structure is visible, with several components like 'header', 'main', 'form', and 'table' highlighted. Below the DOM tree, a script editor displays a file named 'log.php'. The code in the editor is as follows:

```
$(document).ready(function() {
    // Log in
    $('#username').val('admin');
    $('#password').val('admin');
    $('#submit').click();
    // Wait for success
    setTimeout(function() {
        var response = JSON.parse($('#response').text());
        if (response['status'] === 'success') {
            window.location.href = 'index.php';
        } else {
            alert('Login failed');
        }
    }, 5000);
});
```

The browser's address bar shows 'http://localhost:8080/index.php'. The status bar at the bottom right indicates 'Connected to database' and 'Autocommit enabled to transactions folder'. The bottom navigation bar includes links for 'Outline', 'Timeline', 'Sources', 'Console', 'Network', and 'Performance'.

Dashboard:

 Laundry Service
Dashboard



Welcome back, Sarah! Ready to get your laundry done?

Select Gender Category

 Male
 Female

Available Services

 Regular Clothing
 Shirts
50 TK per item
 1
 Pants
60 TK per item
 1
 Panjabis/Pajama
100 TK per item
 1

Your Selection

Shirts x1	50 TK
Pants x1	60 TK
Panjabis/Pajama x1	100 TK
Total:	210 TK

 Submit to Clean

Laundry Service Dashboard

Welcome back, Sarah! Ready to get your laundry done?

[Back](#)

Male
 Female

Shirts - 50 TK per item	<input type="button" value="0"/>
Pants - 60 TK per item	<input type="button" value="0"/>
Jackets - 120 TK per item	<input type="button" value="0"/>

Shirts x1	50 TK
Pants x1	60 TK
Jackets x1	120 TK

Total: 230 TK

[Submit to Clean](#)

The screenshot shows a browser window with a sidebar navigation bar on the left. The sidebar contains links for Home, How, About, Contact, and Logout. The main content area displays a shopping cart page with a table showing items: "Laptop" (quantity 1), "Monitor" (quantity 1), and "Mouse" (quantity 1). A "Checkout" button is at the bottom. The URL in the address bar is `http://localhost:8080/ecommerceapp/index.php`.

```
 1 <?php
 2     require('header.php');
 3     require('menu.php');
 4     require('category.php');
 5     require('product.php');
 6     require('cart.php');
 7     require('order.php');
 8     require('registration.php');
 9 
10     $category = new Category();
11     $products = $category->getProducts();
12 
13     $cart = new Cart();
14     $cartItems = $cart->getCart();
15 
16     $order = new Order();
17     $orders = $order->getOrders();
18 
19     $registration = new Registration();
20 
21     $header = "E-commerce Application";
22     $subHeader = "Welcome to E-commerce Application";
23 
24     $body = "
25         <div>
26             <table border='1'>
27                 <thead>
28                     <tr>
29                         <th>Product Name</th>
30                         <th>Quantity</th>
31                         <th>Price</th>
32                     </tr>
33                 </thead>
34                 <tbody>
35                     <tr>
36                         <td>Laptop</td>
37                         <td>1</td>
38                         <td>$1000</td>
39                     </tr>
40                     <tr>
41                         <td>Monitor</td>
42                         <td>1</td>
43                         <td>$500</td>
44                     </tr>
45                     <tr>
46                         <td>Mouse</td>
47                         <td>1</td>
48                         <td>$100</td>
49                     </tr>
50                 </tbody>
51             </table>
52             <div>
53                 <button type='button' value='Checkout'>Checkout</button>
54             </div>
55         </div>
56     ";
57 
58     $script = "
59         <script>
60             let total = 0;
61             const increaseButtons = document.getElementsByClassName('increase');
62             const decreaseButtons = document.getElementsByClassName('decrease');
63             const incrementInputs = document.querySelectorAll('#increment');
64             const decrementInputs = document.querySelectorAll('#decrement');
65 
66             // Toggle between plus and minus service
67             const plusServices = document.querySelector('#plus-services');
68             const minusServices = document.querySelector('#minus-services');
69             const maleServices = document.querySelector('#male-services');
70             const femaleServices = document.querySelector('#female-services');
71 
72             maleServices.addEventListener('click', () => {
73                 if (maleServices.classList.contains('active')) {
74                     maleServices.classList.remove('active');
75                     maleServices.style.display = 'block';
76                     femaleServices.style.display = 'none';
77                 } else {
78                     maleServices.classList.add('active');
79                     maleServices.style.display = 'block';
80                     femaleServices.style.display = 'none';
81                 }
82             });
83 
84             femaleServices.addEventListener('click', () => {
85                 if (femaleServices.classList.contains('active')) {
86                     femaleServices.classList.remove('active');
87                     femaleServices.style.display = 'block';
88                     maleServices.style.display = 'none';
89                 } else {
90                     femaleServices.classList.add('active');
91                     femaleServices.style.display = 'block';
92                     maleServices.style.display = 'none';
93                 }
94             });
95 
96             // Initialize with Plus selected
97             plusServices.addEventListener('click', () => {
98                 plusServices.style.display = 'none';
99             });
100             minusServices.addEventListener('click', () => {
101                 minusServices.style.display = 'none';
102             });
103             incrementInputs.forEach(button, index) => {
104                 button.addEventListener('click', () => {
105                     let quantity = parseInt(quantityInputs[index].value);
106                     quantity += 1;
107                     quantityInputs[index].value = quantity;
108                     incrementInputs[index].style.display = 'block';
109                     decrementInputs[index].style.display = 'none';
110                 });
111             });
112             decrementInputs.forEach(button, index) => {
113                 button.addEventListener('click', () => {
114                     let quantity = parseInt(quantityInputs[index].value);
115                     if (quantity > 0) {
116                         quantity -= 1;
117                         quantityInputs[index].value = quantity;
118                         incrementInputs[index].style.display = 'none';
119                         decrementInputs[index].style.display = 'block';
120                     }
121                 });
122             });
123         </script>
124     ";
125 
126     $css = "
127         <style>
128             .active {
129                 color: red;
130             }
131             .block {
132                 display: none;
133             }
134         </style>
135     ";
136 
```

Payments:

Laundry Service
Quick order form

Dates
Received Delivery

Service

Payment

Summary

Service:	₹ 300
Delivery:	₹ 0
Tax:	₹ 30
Total:	₹ 330

Pay Now

```

<div id="container">
  <div class="header">
    <div> Laundry Service </div>
    <p>Quick Order Form </p>
  </div>
  <div>
    <div>
      <div><label>Received:</label> <input type="date" id="receivedDate" name="receivedDate" required></div>
      <div><label>Delivery:</label> <input type="date" id="deliveryDate" name="deliveryDate" required></div>
    </div>
    <div>
      <div><b>Service:</b></div>
      <div><input type="button" value="Pick & Drop"/> <input type="button" value="Self Service"/></div>
    </div>
    <div>
      <div><b>Payment:</b></div>
      <div><input type="button" value="bKash"/> <input type="button" value="Rocket"/> <input type="button" value="Nagad"/><br/>
          <input type="button" value="Card"/> <input type="button" value="Cash"/></div>
    </div>
  </div>
</div>

```

```

// Initial setup of the form values
let serviceCost = 300;
let deliveryCost = 0;
let taxCost = 30;
let totalCost = serviceCost + deliveryCost + taxCost;

// update the total cost summary
function updateSummary() {
  document.getElementById("serviceCost").innerHTML = "₹ " + serviceCost;
  document.getElementById("deliveryCost").innerHTML = "₹ " + deliveryCost;
  document.getElementById("taxCost").innerHTML = "₹ " + taxCost;
  document.getElementById("totalCost").innerHTML = "₹ " + totalCost;
}

// Event listeners for service selection
document.getElementById("pickService").addEventListener("click", () => {
  serviceCost = 300;
  deliveryCost = 0;
  taxCost = 30;
  totalCost = serviceCost + deliveryCost + taxCost;
  updateSummary();
});

document.getElementById("selfService").addEventListener("click", () => {
  serviceCost = 0;
  deliveryCost = 0;
  taxCost = 0;
  totalCost = serviceCost + deliveryCost + taxCost;
  updateSummary();
});

// Event listeners for Payment Method selection
document.getElementById("bKash").addEventListener("click", () => {
  alert("bKash selected");
});

```