



American International University- Bangladesh(AIUB)

Department of Computer Science

Faculty of Science Technology

Summer 2023-24

Advance Database Management System

Section: A

Project Title: Office Recreational Fund Management

Submitted by:

Name	ID	Contribution
Sumaiya Tasnim	21-45583-3	33%
Nishat Afla	21-45574-3	33%
Mahmud Al Ashiq	21-45010-2	33%

Contents

- | | |
|---------------------------------------|---------|
| 1. Introduction..... | Page 2 |
| 2. Scenario Description..... | Page 2 |
| 3. ER Diagram..... | Page 3 |
| 4. Normalization..... | Page 4 |
| 5. Table Creation..... | Page 10 |
| 6. Data Insertion..... | Page 19 |
| 7. Query Writing: Basic PL/SQL..... | Page 27 |
| 8. Query Writing: Advance PL/SQL..... | Page 44 |
| 9. Conclusion..... | Page 62 |

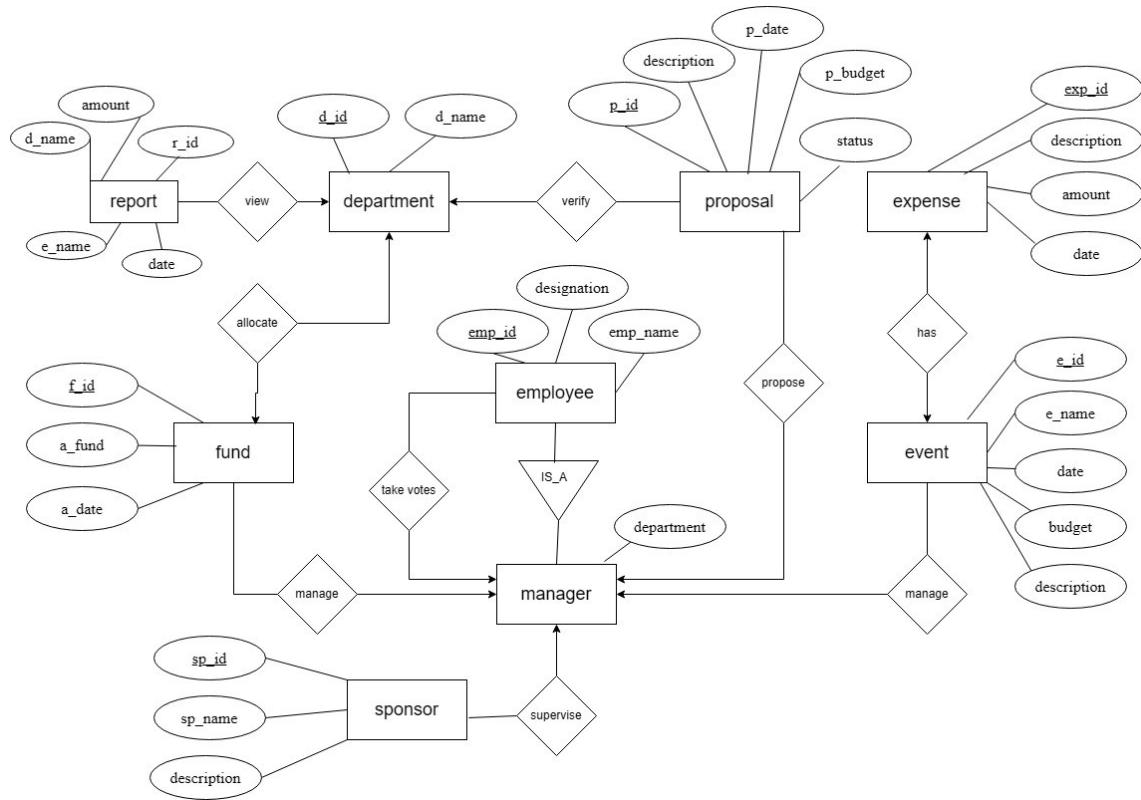
Introduction:

The office's Recreational Fund Management team helps organize fun events for employees using funds that are set aside each month or year. These events, like team outings or parties, have specific details such as event name, budget, and date, and each one has certain expenses, like food or decorations. Different departments in the company contribute money to the recreational fund, and the office manager oversees all the funds. The manager, who is also an employee, takes care of managing the money and making sure it's used wisely for events. Therefore, this system helps the office organize enjoyable activities while keeping track of all the money that's used for them.

Scenario Description:

The office recreational fund management organizes activities for employees. In an office, the Recreational Fund Management contributes monthly and yearly recreational funds to the events. Each event is identified by an event ID, name, date, budget, and description. An event has an expense. Each event expense has an expense ID, description, amount, and date. A department allocates a fund as a recreational fund. The office manager can manage multiple funds. Every fund has a fund ID, allocated fund, and allocation date. Each department has a department ID and name. The office manager is a special type of employee, and the department managed by the manager is stored in the database. Each employee has an employee ID, name, department, and designation. The office manager can supervise all the sponsors. Every sponsor has a sponsor ID, description, and name. The manager takes votes from the company's employees on decisions such as location and activities in a recreational event and can propose fund proposals to the department. Departments verify the proposals. Each proposal has a proposal ID, description, proposed date, and proposed budget and status, indicating whether the proposal is accepted or rejected. A department can view reports that provide a total overview of all the recreational funds. These reports include a report ID, department name, event name, amounts, and dates.

ER Diagram:



Normalization

Manage_fund:

UNF: f_id, a_fund, a_date, emp_id, emp_name, designation, department

1NF: f_id, a_fund, a_date, emp_id, emp_name, designation, department

2NF:

1st: f_id, a_fund, a_date

2nd: emp_id, emp_name, designation, department

3NF:

1st: f_id, a_fund, a_date

2nd: emp_id, emp_name, designation, department

Table:

1. f_id, a_fund, a_date, **emp_id**
2. emp_id, emp_name, designation, department

Has:

UNF: exp_id, description, amount, date, e_id, e_name, date, budget, description

1NF: exp_id, description, amount, date, e_id, e_name, date, budget, description

2NF:

1st: exp_id, description, amount, date

2nd: e_id, e_name, date, budget, description

3NF:

1st: exp_id, description, amount, date

2nd: e_id, e_name, date, budget, description

Table:

1. exp_id, description, amount, date
2. e_id, e_name, date, budget, description, exp_id

Manage_event:

UNF: emp_id, emp_name, designation, department, e_id, e_name, date, budget, description

1NF: emp_id, emp_name, designation, department, e_id, e_name, date, budget, description

2NF:

1st: emp_id, emp_name, designation, department

2nd: e_id, e_name, date, budget, description

3NF:

1st: emp_id, emp_name, designation, department

2nd: e_id, e_name, date, budget, description

Table:

1. emp_id, emp_name, designation, department
2. e_id, e_name, date, budget, description, **emp_id**

Supervise:

UNF: sp_id, description, sp_name, emp_id, emp_name, department, designation

1NF: sp_id, description, sp_name, emp_id, emp_name, department, designation

2NF:

1st: sp_id, description, sp_name

2nd: emp_id, emp_name, department, designation

3NF:

1st: sp_id, description, sp_name

2nd: emp_id, emp_name, department, designation

Table:

1. sp_id, description, sp_name, **emp_id**
2. emp_id, emp_name, department, designation

\

Propose:

UNF: p_id, description, p_date, p_budget, status, emp_id, emp_name, department, designation

1NF: p_id, description, p_date, p_budget, status, emp_id, emp_name, department, designation

2NF:

1st: p_id, description, p_date, p_budget, status

2nd: emp_id, emp_name, department, designation

3NF:

1st: p_id, description, p_date, p_budget, status

2nd: emp_id, emp_name, department, designation

Table:

1. p_id, description, p_date, p_budget, status, **emp_id**

2. emp_id, emp_name, department, designation

Allocate:

UNF: f_id, a_fund, a_date, d_id, d_name

1NF: f_id, a_fund, a_date, d_id, d_name

2NF:

1st: f_id, a_fund, a_date

2nd: d_id, d_name

3NF:

1st: f_id, a_fund, a_date

2nd: d_id, d_name

Table:

1. f_id, a_fund, a_date

2. d_id, d_name, **f_id**

Verify:

UNF: p_id, description, p_date, p_budget, status, d_id, d_name

1NF: p_id, description, p_date, p_budget, status, d_id, d_name

2NF:

1st: p_id, description, p_date, p_budget, status

2nd: d_id, d_name

3NF:

1st: p_id, description, p_date, p_budget, status

2nd: d_id, d_name

Table:

1. p_id, description, p_date, p_budget, status, **d_id**
2. d_id, d_name

View:

UNF: r_id, amount, date, d_name, e_name, d_id, d_name

1NF: r_id, amount, date, d_name, e_name, d_id, d_name

2NF:

1st: r_id, amount, date, d_name, e_name

2nd: d_id, d_name

3NF:

1st: r_id, amount, date, d_name, e_name

2nd: d_id, d_name

Table:

1. r_id, amount, date, d_name, e_name, **d_id**
2. d_id, d_name

Take votes:

UNF: emp_id, emp_name, designation, emp_id, emp_name, designation, department

1NF: emp_id, emp_name, designation, emp_id, emp_name, designation, department

2NF:

1st: emp_id, emp_name, designation

2nd: emp_id, emp_name, designation, department

3NF:

1st: emp_id, emp_name, designation

2nd: emp_id, emp_name, designation, department

Table:

1. emp_id, emp_name, designation, **emp_id (manager)**
2. emp_id, emp_name, designation, department

Table:

1. **f_id, a_fund, a_date, emp_id**
2. **emp_id, emp_name, designation, department**
3. **exp_id, description, amount, date**
4. **e_id, e_name, date, budget, description, exp_id**
5. **emp_id, emp_name, designation, department**
6. **e_id, e_name, date, budget, description, emp_id**
7. **sp_id, description, sp_name, emp_id**
8. **emp_id, emp_name, department, designation**
9. **p_id, description, p_date, p_budget, status, emp_id**
10. **emp_id, emp_name, department, designation**
11. **f_id, a_fund, a_date**
12. **d_id, d_name, f_id**
13. **p_id, description, p_date, p_budget, status, d_id**
14. **d_id, d_name**
15. **r_id, amount, date, d_name, e_name, d_id**
16. **d_id, d_name**
17. **emp_id, emp_name, designation, emp_id (manager)**
18. **emp_id, emp_name, designation, department**

Final tables:

1. exp_id, description, amount, date
2. emp_id, emp_name, designation, **emp_id (manager)**
3. emp_id, emp_name, designation, department
4. f_id, a_fund, a_date, **emp_id (manager)**
5. e_id, e_name, date, budget, description, exp_id, **emp_id (manager)**
6. sp_id, description, sp_name, **emp_id (manager)**
7. p_id, description, p_date, p_budget, status, **emp_id (manager)**, d_id
8. d_id, d_name, **f_id**
9. r_id, amount, r_date, d_name, e_name, **d_id**

User Creation:

```
CREATE USER adbms_project IDENTIFIED BY admin;  
GRANT UNLIMITED TABLESPACE TO adbms_project ;  
GRANT CREATE SESSION TO adbms_project ;  
GRANT CREATE TABLE TO adbms_project ;  
GRANT CREATE VIEW TO adbms_project ;  
GRANT CREATE SEQUENCE TO adbms_project ;
```

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

```

CREATE USER adms_project IDENTIFIED BY admin;
GRANT UNLIMITED TABLESPACE TO adms_project ;
GRANT CREATE SESSION TO adms_project ;
GRANT CREATE TABLE TO adms_project ;
GRANT CREATE VIEW TO adms_project ;
GRANT CREATE SEQUENCE TO adms_project ;
```

The results pane shows the message "Statement processed." and a duration of "0.00 seconds". The status bar at the bottom right indicates the application version is "Application Express 2.1.0.00.39" and the copyright is "Copyright © 1999, 2006, Oracle. All rights reserved."

Table Creation:

expense		
Column Name	Data type	Constraint
exp_id	Number(4)	Primary Key
description	Varchar2(100)	Not null
Amount	Number	Not null
date	Date	Not null

```

CREATE TABLE expense (
    exp_id      NUMBER(4) PRIMARY KEY,
    description  VARCHAR2(100) NOT NULL,
    amount      NUMBER NOT NULL,
    expense_date DATE NOT NULL
);
DESC expense;
```

User SCOTT

Home > SQL > SQL Commands

Autocommit

```
CREATE TABLE expense (
    exp_id      NUMBER(4) PRIMARY KEY,
    description VARCHAR2(100) NOT NULL,
    amount      NUMBER NOT NULL,
    expense_date DATE NOT NULL
);

DESC expense;
```

Results Explain Describe Saved SQL History

Object Type TABLE Object EXPENSE

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EXPENSE	EXP_ID	Number	-	4	0	1	-	-	-
	DESCRIPTION	VARCHAR2	100	-	-	-	-	-	-
	AMOUNT	Number	-	-	-	-	-	-	-
	EXPENSE_DATE	Date	7	-	-	-	-	-	-

1 - 4



employee		
Column Name	Data type	Constraint
emp_id	Number(4)	Primary Key
emp_name	VARCHAR2(30)	Not null
designation	VARCHAR2(10)	Not null
manager_id	Number(4)	Foreign key

```
CREATE TABLE employee (
    emp_id      NUMBER(4) PRIMARY KEY,
    emp_name    VARCHAR2(30) NOT NULL,
    designation VARCHAR2(10) NOT NULL,
    manager_id  NUMBER(4),
    CONSTRAINT fk_manager FOREIGN KEY (manager_id) REFERENCES
    manager(emp_id)
);DESC employee;
```

SQL Commands

ORACLE Database Express Edition

User: SCOTT

Home > SQL > SQL Commands

Autocommit

```
CREATE TABLE employee (
    emp_id      NUMBER(4) PRIMARY KEY,
    emp_name    VARCHAR2(30) NOT NULL,
    designation VARCHAR2(10) NOT NULL,
    manager_id  NUMBER(4),
    CONSTRAINT fk_manager FOREIGN KEY (manager_id) REFERENCES
manager(emp_id)
);
DESC employee;
```

Results Explain Describe Saved SQL History

Object Type TABLE Object EMPLOYEE

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMPLOYEE	EMP_ID	Number	-	4	0	1	-	-	
	EMP_NAME	Varchar2	30	-	-	-	-	-	
	DESIGNATION	Varchar2	10	-	-	-	-	-	
	MANAGER_ID	Number	-	4	0	-	✓	-	

1 - 4

Type here to search Rain to stop 12:20 PM 7/4/2024

department		
Column Name	Data type	Constraint
d_id	Number(4)	Primary Key
d_name	Varchar2(30)	Not null
f_id	Number(4)	Foreign Key

```
CREATE TABLE department (
d_id NUMBER(4) PRIMARY KEY,
d_name VARCHAR2(30) NOT NULL,
f_id NUMBER(4),
CONSTRAINT fk_fund FOREIGN KEY (f_id) REFERENCES fund(f_id)
);
DESC department;
```

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

```

CREATE TABLE department (
    d_id NUMBER(4) PRIMARY KEY,
    d_name VARCHAR2(30) NOT NULL,
    f_id NUMBER(4),
    CONSTRAINT fk_fund FOREIGN KEY (f_id) REFERENCES fund(f_id)
);
DESC department;

```

Below the code, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected. A table titled "Object Type TABLE Object DEPARTMENT" is displayed, showing the columns and their properties:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPARTMENT	d_id	Number	4	0		1	-	-	
	d_name	Varchar2	30	-		-	-	-	
	f_id	Number	-	4	0	-	✓	-	

At the bottom right, it says "1 - 3".

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



manager		
Column Name	Data type	Constraint
emp_id	Number(4)	Primary Key
emp_name	Varchar2(30)	Not null
designation	Varchar2(10)	Not null
department	Varchar2(10)	Not null

```

CREATE TABLE manager (
    emp_id    NUMBER(4) PRIMARY KEY,
    emp_name  VARCHAR2(30) NOT NULL,
    designation VARCHAR2(10) NOT NULL,
    department VARCHAR2(10) NOT NULL
);
DESC manager;

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The URL is 127.0.0.1:8080/apex/f?p=4500:1003:3363423518004011::NO:::. The page title is ORACLE Database Express Edition. The user is SCOTT. The SQL command entered is:

```
CREATE TABLE manager (
    emp_id NUMBER(4) PRIMARY KEY,
    emp_name VARCHAR2(30) NOT NULL,
    designation VARCHAR2(10) NOT NULL,
    department VARCHAR2(10) NOT NULL
);

DESC manager;
```

Below the command, there are tabs for Results, Explain, Describe, Saved SQL, and History. The Results tab is selected. A table titled "Object Type TABLE Object MANAGER" shows the structure of the manager table:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MANAGER	EMP_ID	Number	-	4	0	1	-	-	
	EMP_NAME	VARCHAR2	30	-	-	-	-	-	
	DESIGNATION	VARCHAR2	10	-	-	-	-	-	
	DEPARTMENT	VARCHAR2	10	-	-	-	-	-	

At the bottom, it says 1 - 4.

Below the interface, the Windows taskbar is visible with various icons and the system tray showing the date and time.

	fund	
Column Name	Data type	Constraint
<u>f_id</u>	Number(4)	Primary Key
a_fund	Number(10)	Not null
a_date	Date	Not null
emp_id	Number(4)	Foreign Key

```
CREATE TABLE fund (
    f_id      NUMBER(4) PRIMARY KEY,
    a_fund    NUMBER NOT NULL,
    a_date    DATE NOT NULL,
    manager_id NUMBER(4),
    CONSTRAINT fk_employee FOREIGN KEY (manager_id) REFERENCES
    employee(emp_id)
```

);

DESC fund;

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

```
CREATE TABLE fund (
    f_id      NUMBER(4) PRIMARY KEY,
    a_fund    NUMBER NOT NULL,
    a_date    DATE NOT NULL,
    manager_id NUMBER(4),
    CONSTRAINT fk_employee FOREIGN KEY (manager_id) REFERENCES employee(emp_id)
);
DESC fund;
```

Below the code, the results section displays the table structure:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
FUND	F_ID	Number	-	4	0	1	-	-	-
	A_FUND	Number	-	-	-	-	-	-	-
	A_DATE	Date	7	-	-	-	-	-	-
	MANAGER_ID	Number	-	4	0	-	✓	-	-

At the bottom, the status bar shows "Application Express 2.1.0.00.39" and "Copyright © 1999, 2008, Oracle. All rights reserved."

event

Column Name	Data type	Constraint
e_id	Number(4)	Primary Key
e_name	Varchar2(30)	Not null
event_date	Date	Not null
budget	Number(10)	Not null
description	Varchar2(100)	Not null
exp_id	Number(4)	Foreign Key
emp_id	Number(4)	Foreign Key

```
CREATE TABLE event (
    e_id      NUMBER(4) PRIMARY KEY,
    e_name    VARCHAR2(30) NOT NULL,
    event_date DATE NOT NULL,
```

```

budget      NUMBER(10) NOT NULL,
description  VARCHAR2(100) NOT NULL,
exp_id       NUMBER(4),
manager_id   NUMBER(4),
CONSTRAINT fk_exp FOREIGN KEY (exp_id) REFERENCES expense(exp_id),
CONSTRAINT fk_emp FOREIGN KEY (manager_id) REFERENCES employee(emp_id));
DESC event;

```

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

```

CREATE TABLE event (
    e_id          NUMBER(4) PRIMARY KEY,
    e_name        VARCHAR2(30) NOT NULL,
    event_date    DATE NOT NULL,
    budget        NUMBER(10) NOT NULL,
    description   VARCHAR2(100) NOT NULL,
    exp_id        NUMBER(4),
    manager_id   NUMBER(4),
    CONSTRAINT fk_exp FOREIGN KEY (exp_id) REFERENCES expense(exp_id),
    CONSTRAINT fk_emp FOREIGN KEY (manager_id) REFERENCES employee(emp_id));
DESC event;

```

Below the code, there is a table titled "Object Type TABLE Object EVENT" showing the columns and their properties:

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EVENT	E_ID	Number	-	4	0	1	-	-	-
	E_NAME	Varchar2	30	-	-	-	-	-	-
	EVENT_DATE	Date	7	-	-	-	-	-	-
	BUDGET	Number	-	10	0	-	-	-	-
	DESCRIPTION	Varchar2	100	-	-	-	-	-	-
	EXP_ID	Number	-	4	0	-	✓	-	-
	MANAGER_ID	Number	-	4	0	-	✓	-	-

At the bottom of the interface, it says "Application Express 2.1.0.0.39" and "Copyright © 1999, 2008, Oracle. All rights reserved."

sponsor		
Column Name	Data type	Constraint
sp_id	Number(4)	Primary Key
description	Varchar2(100)	Not null
sp_name	Varchar2(30)	Not null
emp_id	Number(4)	Foreign Key

```

CREATE TABLE sponsor (
sp_id      NUMBER(4) PRIMARY KEY,
description VARCHAR2(100) NOT NULL,
sp_name    VARCHAR2(30) NOT NULL,
manager_id  NUMBER(4),
CONSTRAINT fk_e FOREIGN KEY (manager_id) REFERENCES employee(emp_id)

```

);

DESC sponsor;

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

```
CREATE TABLE sponsor (
    sp_id      NUMBER(4) PRIMARY KEY,
    description VARCHAR2(100) NOT NULL,
    sp_name     VARCHAR2(30) NOT NULL,
    manager_id   NUMBER(4),
    CONSTRAINT fk_e FOREIGN KEY (manager_id) REFERENCES employee(emp_id)
);
DESC sponsor;
```

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

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SPONSOR	SP_ID	Number	-	4	0	1	-	-	-
	DESCRIPTION	Varchar2	100	-	-	-	-	-	-
	SP_NAME	Varchar2	30	-	-	-	-	-	-
	MANAGER_ID	Number	-	4	0	-	✓	-	-

At the bottom of the interface, there are status indicators: Application Express 2.1.0.00.39, Copyright © 1999, 2008, Oracle. All rights reserved.

proposal

Column Name	Data type	Constraint
p_id	Number(4)	Primary Key
description	Varchar2(100)	Not null
p_date	Date	Not null
p_budget	Number(10)	Not null
status	Varchar2(30)	Not null
emp_id	Number(4)	Foreign Key
d_id	Number(4)	Foreign Key

```
CREATE TABLE proposal (
    p_id      NUMBER(4) PRIMARY KEY,
    description  VARCHAR2(100) NOT NULL,
    p_date      DATE NOT NULL,
```

```

p_budget NUMBER(10) NOT NULL,
status VARCHAR2(30) NOT NULL,
manager_id NUMBER(4),
d_id NUMBER(4),
CONSTRAINT fk_emplo FOREIGN KEY (manager_id) REFERENCES
employee(emp_id),
CONSTRAINT fk_depar FOREIGN KEY (d_id) REFERENCES department(d_id));

```

DESC proposal;

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

```

CREATE TABLE proposal (
    p_id          NUMBER(4) PRIMARY KEY,
    description   VARCHAR2(100) NOT NULL,
    p_date        DATE NOT NULL,
    p_budget      NUMBER(10) NOT NULL,
    status        VARCHAR2(30) NOT NULL,
    manager_id    NUMBER(4),
    d_id          NUMBER(4),
    CONSTRAINT fk_emplo FOREIGN KEY (manager_id) REFERENCES employee(emp_id),
    CONSTRAINT fk_depar FOREIGN KEY (d_id) REFERENCES department(d_id));
DESC proposal;

```

Below the command, the 'DESC' output is shown:

```

Object Type TABLE Object PROPOSAL
Table Column Data Type Length Precision Scale Primary Key Nullable Default Comment
PROPOSAL P_ID Number - 4 0 1 - - -
DESCRIPTION Varchar2 100 - - - - -
P_DATE Date 7 - - - - -
P_BUDGET Number - 10 0 - - - -
STATUS Varchar2 30 - - - - -
MANAGER_ID Number - 4 0 - ✓ - -
D_ID Number - 4 0 - ✓ - -

```

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

report			
Column Name	Data type	Constraint	
r_id	Number(4)	Primary Key	
amount	Number(10)	Not null	
r_date	Date	Not null	
d_name	Varchar2(30)	Not null	
e_name	Varchar2(30)	Not null	
d_id	Number(4)	Foreign Key	

```

CREATE TABLE report (
r_id    NUMBER(4) PRIMARY KEY,
amount  NUMBER(10) NOT NULL,
r_date  DATE NOT NULL,
d_name  VARCHAR2(30) NOT NULL,

```

```

e_name  VARCHAR2(30) NOT NULL,
d_id    NUMBER(4),
CONSTRAINT fk_depart FOREIGN KEY (d_id) REFERENCES department(d_id));
DESC report;

```

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

```

CREATE TABLE report (
    r_id    NUMBER(4) PRIMARY KEY,
    amount  NUMBER(10) NOT NULL,
    r_date  DATE NOT NULL,
    d_name  VARCHAR2(30) NOT NULL,
    e_name  VARCHAR2(30) NOT NULL,
    d_id    NUMBER(4),
    CONSTRAINT fk_depart FOREIGN KEY (d_id) REFERENCES department(d_id)
);

DESC report;

```

The results show the table structure:

Object Type	Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
REPORT	R_ID	Number	-	4	0		1	-	-	-
	AMOUNT	Number	-	10	0		-	-	-	-
	R_DATE	Date	7	-	-		-	-	-	-
	D_NAME	Varchar2	30	-	-		-	-	-	-
	E_NAME	Varchar2	30	-	-		-	-	-	-
	D_ID	Number	-	4	0		-	✓	-	-

1 - 6

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



Data Insertion

Table name: employee

```

INSERT INTO employee VALUES (1001, 'Sumaiya Tasnim', 'Developer', 1007);
INSERT INTO employee VALUES (1002, 'Nishat Afla', 'Developer', 1009);
INSERT INTO employee VALUES (1003, 'Mahmud Al Ashiq', 'Engineer', 1007);
INSERT INTO employee VALUES (1004, 'Chondrima Kumari', 'Developer', 1010);
INSERT INTO employee VALUES (1005, 'Kamal Hasib', 'Analyst', 1008);
INSERT INTO employee VALUES (1006, 'Rodoshie Reehean', 'Engineer', 1010);
INSERT INTO employee VALUES (1007, 'Baria Ifteen', 'Manager', NULL);
INSERT INTO employee VALUES (1008, 'Tahsib Alam', 'Manager', NULL);
INSERT INTO employee VALUES (1009, 'Lance Florida', 'Manager', NULL);

```

INSERT INTO employee VALUES (1010, 'Jessia Islam', 'Manager', NULL);

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

```
select * from employee
```

The results are displayed in a table:

EMP_ID	EMP_NAME	DESIGNATION	MANAGER_ID
1001	Sumaiya Tasnim	Developer	1007
1002	Nishat Afia	Developer	1009
1003	Mahmud Al Ashiq	Engineer	1007
1004	Chondrima Kumar	Developer	1010
1005	Kamal Hasib	Analyst	1008
1006	Rodostha Reheen	Engineer	1010
1007	Baria Ifteen	Manager	-
1008	Tahsib Alam	Manager	-
1009	Lance Florida	Manager	-
1010	Jessia Islam	Manager	-

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

Table name: manager

INSERT INTO manager VALUES (1007, 'Baria Ifteen', 'Manager', 'IT');

INSERT INTO manager VALUES (1008, 'Tahsib Alam', 'Manager', 'Networking');

INSERT INTO manager VALUES (1009, 'Lance Florida', 'Manager', 'Finance');

INSERT INTO manager VALUES (1010, 'Jessia Islam', 'Manager', 'HR');

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

```
select * from manager
```

The results are displayed in a table:

EMP_ID	EMP_NAME	DESIGNATION	DEPARTMENT
1007	Baria Ifteen	Manager	IT
1008	Tahsib Alam	Manager	Networking
1009	Lance Florida	Manager	Finance
1010	Jessia Islam	Manager	HR

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

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

Table name: expense

```
INSERT INTO expense VALUES (3001, 'Picnic for new years eve', 8000, TO_DATE('2024-01-01', 'YYYY-MM-DD'));
```

```
INSERT INTO expense VALUES (3002, 'Birthday Surprise to Sumaiya', 3000, TO_DATE('2024-01-10', 'YYYY-MM-DD'));
```

```
INSERT INTO expense VALUES (3003, 'Dinner Outing to a Banani restaurant', 6500, TO_DATE('2024-02-17', 'YYYY-MM-DD'));
```

```
INSERT INTO expense VALUES (3004, 'Sudden Lunch plan to nearby cafe', 1000, TO_DATE('2024-02-20', 'YYYY-MM-DD'));
```

```
INSERT INTO expense VALUES (3005, 'Dinner outing for company anniversary', 8500, TO_DATE('2024-03-11', 'YYYY-MM-DD'));
```

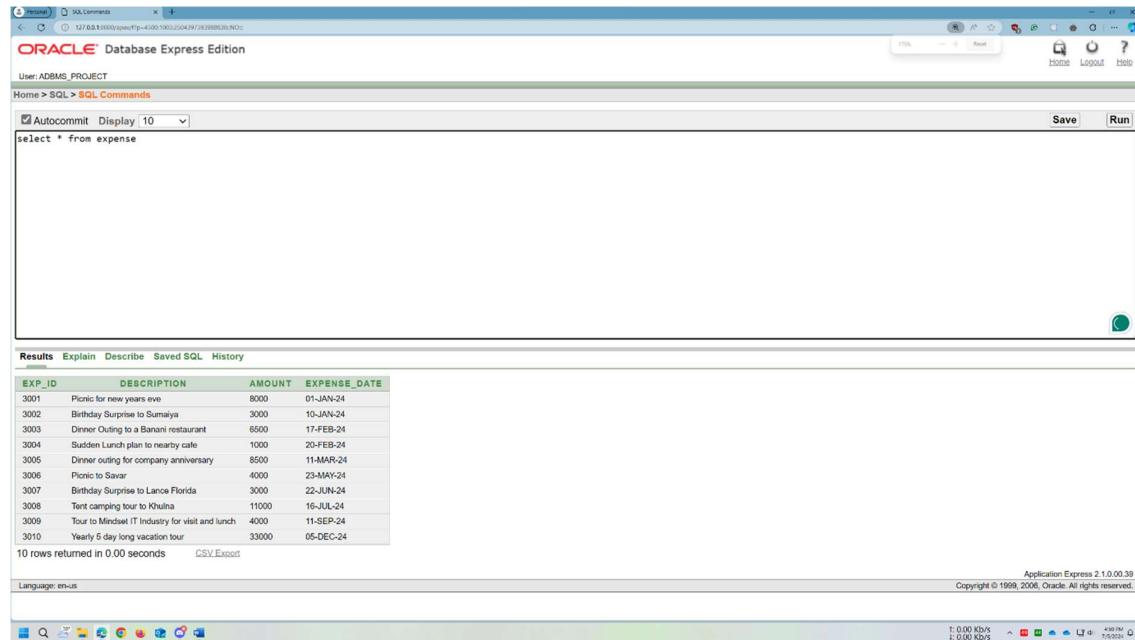
```
INSERT INTO expense VALUES (3006, 'Picnic to Savar', 4000, TO_DATE('2024-05-23', 'YYYY-MM-DD'));
```

```
INSERT INTO expense VALUES (3007, 'Birthday Surprise to Lance Florida', 3000, TO_DATE('2024-06-22', 'YYYY-MM-DD'));
```

```
INSERT INTO expense VALUES (3008, 'Tent camping tour to Khulna', 11000, TO_DATE('2024-07-16', 'YYYY-MM-DD'));
```

```
INSERT INTO expense VALUES (3009, 'Tour to Mindset IT Industry for visit and lunch', 4000, TO_DATE('2024-09-11', 'YYYY-MM-DD'));
```

```
INSERT INTO expense VALUES (3010, 'Yearly 5 day long vacation tour', 33000, TO_DATE('2024-12-05', 'YYYY-MM-DD'));
```



The screenshot shows the Oracle Database Express Edition interface. The SQL Commands window displays the following SQL query and its results:

```
select * from expense
```

EXP_ID	DESCRIPTION	AMOUNT	EXPENSE_DATE
3001	Picnic for new years eve	8000	01-JAN-24
3002	Birthday Surprise to Sumaiya	3000	10-JAN-24
3003	Dinner Outing to a Banani restaurant	6500	17-FEB-24
3004	Sudden Lunch plan to nearby cafe	1000	20-FEB-24
3005	Dinner outing for company anniversary	8500	11-MAR-24
3006	Picnic to Savar	4000	23-MAY-24
3007	Birthday Surprise to Lance Florida	3000	22-JUN-24
3008	Tent camping tour to Khulna	11000	16-JUL-24
3009	Tour to Mindset IT Industry for visit and lunch	4000	11-SEP-24
3010	Yearly 5 day long vacation tour	33000	05-DEC-24

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

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

Language: en-us

Table name: fund

```
INSERT INTO fund VALUES (4001, 15000, TO_DATE('2024-01-01','YYYY-MM-DD'),  
1007);
```

```
INSERT INTO fund VALUES (4002, 10000, TO_DATE('2024-01-03','YYYY-MM-DD'),  
1008);
```

```
INSERT INTO fund VALUES (4003, 8000, TO_DATE('2024-01-04','YYYY-MM-DD'),  
1009);
```

```
INSERT INTO fund VALUES (4004, 9000, TO_DATE('2024-01-06','YYYY-MM-DD'),  
1010);
```

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

```
select * from fund
```

The results section displays the following data:

F_ID	A_FUND	A_DATE	MANAGER_ID
4001	15000	01-JAN-24	1007
4002	10000	03-JAN-24	1008
4003	8000	04-JAN-24	1009
4004	9000	06-JAN-24	1010

4 rows returned in 0.00 seconds

Table name: department

```
INSERT INTO department VALUES (5001,'IT', 4001);
```

```
INSERT INTO department VALUES (5002,'Finance', 4002);
```

```
INSERT INTO department VALUES (5003,'Networking', 4003);
```

```
INSERT INTO department VALUES (5004,'HR', 4004);
```

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

```
select * from department
```

The results section displays the following data:

D_ID	D_NAME	F_ID
5001	IT	4001
5002	Finance	4002
5003	Networking	4003
5004	HR	4004

4 rows returned in 0.00 seconds

Table name: event

```
INSERT INTO event VALUES (6001,'New Year Party', TO_DATE('2024-01-01', 'YYYY-MM-DD'),8000,'Picnic done in Kolabagan', 3001, 1007);
```

```
INSERT INTO event VALUES (6002,'Birthday Party', TO_DATE('2024-01-10', 'YYYY-MM-DD'),3000,'Surprise to a IT dept member', 3002, 1007);
```

```
INSERT INTO event VALUES (6004,'Cafe outing', TO_DATE('2024-02-24', 'YYYY-MM-DD'),1000,'Cafe foods supplied', 3004, 1009);
```

```
INSERT INTO event VALUES (6005,'Anniversary celebration', TO_DATE('2024-03-24', 'YYYY-MM-DD'),8500,'Anniversary of the company celebrated', 3005, 1010);
```

```
INSERT INTO event VALUES (6006,'Picnic', TO_DATE('2024-01-10', 'YYYY-MM-DD'),4000,'Savar picnic in early morning', 3006, 1008);
```

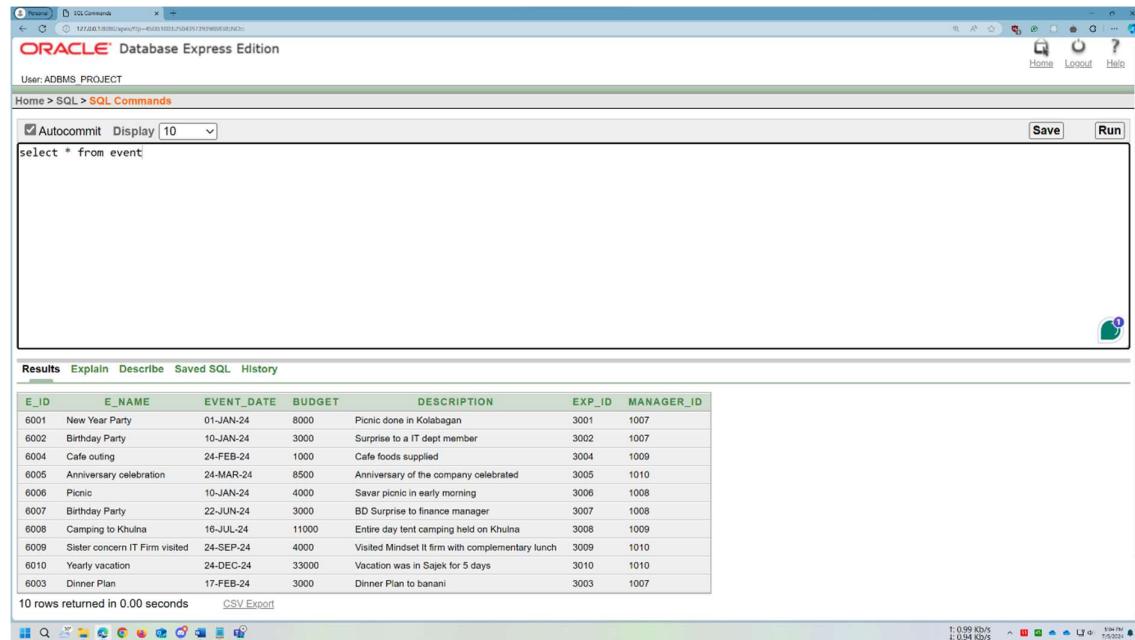
```
INSERT INTO event VALUES (6007,'Birthday Party', TO_DATE('2024-06-22', 'YYYY-MM-DD'),3000,'BD Surprise to finance manager', 3007,1008);
```

```
INSERT INTO event VALUES (6008,'Camping to Khulna', TO_DATE('2024-07-16', 'YYYY-MM-DD'),11000,'Entire day tent camping held on Khulna', 3008, 1009);
```

```
INSERT INTO event VALUES (6009,'Sister concern IT Firm visited', TO_DATE('2024-09-24', 'YYYY-MM-DD'),4000,'Visited Mindset It firm with complementary lunch', 3009,1010);
```

```
INSERT INTO event VALUES (6010,'Yearly vacation', TO_DATE('2024-12-24', 'YYYY-MM-DD'),33000,'Vacation was in Sajek for 5 days', 3010, 1010);
```

```
INSERT INTO event VALUES (6003,'Dinner Plan', TO_DATE('2024-02-17', 'YYYY-MM-DD'),3000,'Dinner Plan to banani', 3003, 1007);
```



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

```
select * from event;
```

The Results tab displays the following data:

E_ID	E_NAME	EVENT_DATE	BUDGET	DESCRIPTION	EXP_ID	MANAGER_ID
6001	New Year Party	01-JAN-24	8000	Picnic done in Kolabagan	3001	1007
6002	Birthday Party	10-JAN-24	3000	Surprise to a IT dept member	3002	1007
6004	Cafe outing	24-FEB-24	1000	Cafe foods supplied	3004	1009
6005	Anniversary celebration	24-MAR-24	8500	Anniversary of the company celebrated	3005	1010
6006	Picnic	10-JAN-24	4000	Savar picnic in early morning	3006	1008
6007	Birthday Party	22-JUN-24	3000	BD Surprise to finance manager	3007	1008
6008	Camping to Khulna	16-JUL-24	11000	Entire day tent camping held on Khulna	3008	1009
6009	Sister concern IT Firm visited	24-SEP-24	4000	Visited Mindset It firm with complementary lunch	3009	1010
6010	Yearly Vacation	24-DEC-24	33000	Vacation was in Sajek for 5 days	3010	1010
6003	Dinner Plan	17-FEB-24	3000	Dinner Plan to banani	3003	1007

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

Table name: sponsor

INSERT INTO sponsor VALUES (7001,'New Year Party for the firm', 'Steven Gadgets', 1007);

INSERT INTO sponsor VALUES (7002,'Birthday Party thrown for an employee from IT dept', 'Pran Ltd', 1007);

INSERT INTO sponsor VALUES (7003,'Dinner to a banani restaurant named bao', 'FIC Insurance', 1009);

INSERT INTO sponsor VALUES (7004,'Cafe outing to cha and chill', 'Pran Ltd', 1010);

INSERT INTO sponsor VALUES (7005,'Company anniversary expenses for gifts and dinner', 'Dolby Company', 1008);

INSERT INTO sponsor VALUES (7006,'Savar outing picnic', 'Dolby Company', 1008);

INSERT INTO sponsor VALUES (7007,'Birthday Party thrown for a manager', 'Steven Gadgets', 1009);

INSERT INTO sponsor VALUES (7008,'Camping on Khulna and expenses on tickets, foods, tents and supplies', 'Micro Supplies Ltd', 1010);

INSERT INTO sponsor VALUES (7009,'Sister Concern company visit', 'FIC Insurance', 1010);

INSERT INTO sponsor VALUES (7010,'5 day vacation to sajek and costs of transportation, hotels, recreation', 'Dolby Company', 1007);

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

```
select * from sponsor
```

The results are displayed in a table:

SP_ID	DESCRIPTION	SP_NAME	MANAGER_ID
7001	New Year Party for the firm	Steven Gadgets	1007
7002	Birthday Party thrown for an employee from IT dept	Pran Ltd	1007
7003	Dinner to a banani restaurant named bao	FIC Insurance	1009
7004	Cafe outing to cha and chill	Pran Ltd	1010
7005	Company anniversary expenses for gifts and dinner	Dolby Company	1008
7006	Savar outing picnic	Dolby Company	1008
7007	Birthday Party thrown for a manager	Steven Gadgets	1009
7008	Camping on Khulna and expenses on tickets, foods, tents and supplies	Micro Supplies Ltd	1010
7009	Sister Concern company visit	FIC Insurance	1010
7010	5 day vacation to sajek and costs of transportation, hotels, recreation	Dolby Company	1007

10 rows returned in 0.02 seconds

Table name: proposal

```
INSERT INTO proposal VALUES (8001,'New Year Party', TO_DATE('2023-12-05',
'YYYY-MM-DD'),8000,'completed', 1007, 5001);
```

```
INSERT INTO proposal VALUES (8002,'Birthday surprise to employee', TO_DATE('2024-
01-02', 'YYYY-MM-DD'),3000,'completed', 1007, 5001);
```

```
INSERT INTO proposal VALUES (8003,'Dinner outing', TO_DATE('2024-01-31', 'YYYY-
MM-DD'),3000,'completed', 1009, 5003);
```

```
INSERT INTO proposal VALUES (8004,'Cafe outing', TO_DATE('2024-02-20', 'YYYY-
MM-DD'),1000,'completed', 1010, 5004);
```

```
INSERT INTO proposal VALUES (8005,'Company Anniversary', TO_DATE('2024-03-01',
'YYYY-MM-DD'),8500,'completed', 1008, 5002);
```

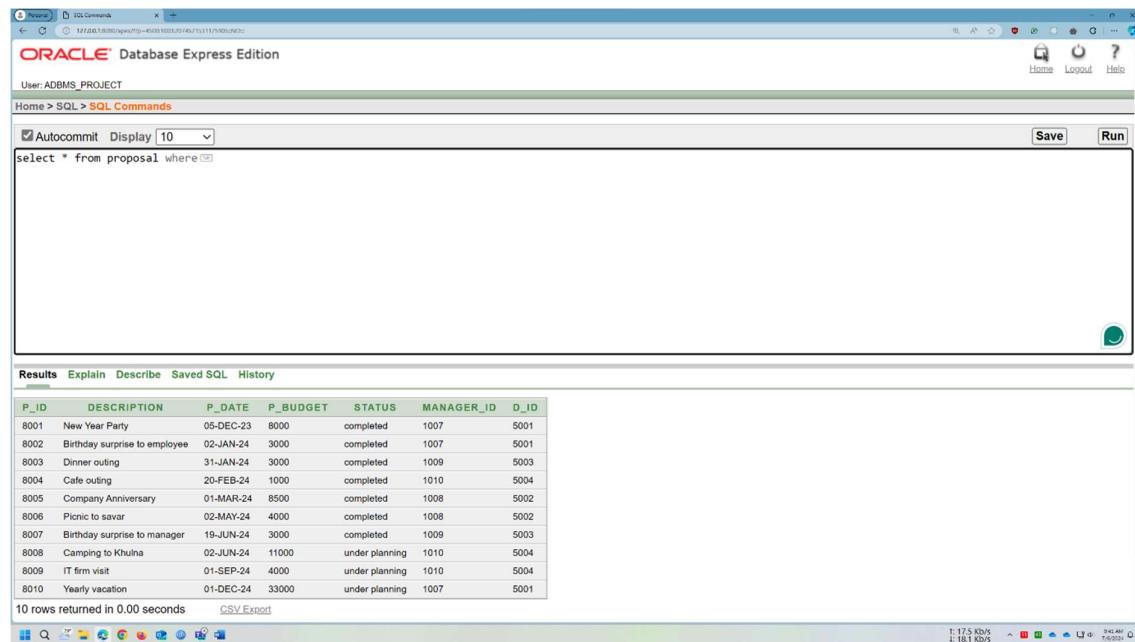
```
INSERT INTO proposal VALUES (8006,'Picnic to savar', TO_DATE('2024-05-02', 'YYYY-
MM-DD'),4000,'completed', 1008, 5002);
```

```
INSERT INTO proposal VALUES (8007,'Birthday surprise to manager', TO_DATE('2024-
06-19', 'YYYY-MM-DD'),3000,'completed', 1009, 5003);
```

```
INSERT INTO proposal VALUES (8008,'Camping to Khulna', TO_DATE('2024-06-02',
'YYYY-MM-DD'),11000,'under planning',1010, 5004);
```

```
INSERT INTO proposal VALUES (8009,'IT firm visit', TO_DATE('2024-09-01', 'YYYY-
MM-DD'),4000,'under planning', 1010, 5004);
```

```
INSERT INTO proposal VALUES (8010,'Yearly vacation', TO_DATE('2024-12-01',
'YYYY-MM-DD'),33000,'under planning', 1007, 5001);
```



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

```
select * from proposal where id
```

P_ID	DESCRIPTION	P_DATE	P_BUDGET	STATUS	MANAGER_ID	D_ID
8001	New Year Party	05-DEC-23	8000	completed	1007	5001
8002	Birthday surprise to employee	02-JAN-24	3000	completed	1007	5001
8003	Dinner outing	31-JAN-24	3000	completed	1009	5003
8004	Cafe outing	20-FEB-24	1000	completed	1010	5004
8005	Company Anniversary	01-MAR-24	8500	completed	1008	5002
8006	Picnic to savar	02-MAY-24	4000	completed	1008	5002
8007	Birthday surprise to manager	19-JUN-24	3000	completed	1009	5003
8008	Camping to Khulna	02-JUN-24	11000	under planning	1010	5004
8009	IT firm visit	01-SEP-24	4000	under planning	1010	5004
8010	Yearly vacation	01-DEC-24	33000	under planning	1007	5001

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

Table name: report

```
INSERT INTO report VALUES (9001, 8000, TO_DATE('2023-01-01', 'YYYY-MM-DD'),'IT', 'New year celebration', 5001);
```

```
INSERT INTO report VALUES (9002, 3000, TO_DATE('2023-01-10', 'YYYY-MM-DD'),'IT', 'BD celebration', 5001);
```

```
INSERT INTO report VALUES (9003, 3000, TO_DATE('2023-01-01', 'YYYY-MM-DD'),'Networking', 'Dinner plan', 5003);
```

```
INSERT INTO report VALUES (9004, 1000, TO_DATE('2023-02-24', 'YYYY-MM-DD'),'IT', 'Cafe outing', 5004);
```

```
INSERT INTO report VALUES (9005, 8500, TO_DATE('2023-01-01', 'YYYY-MM-DD'),'Finance', 'Anniversary celebration', 5002);
```

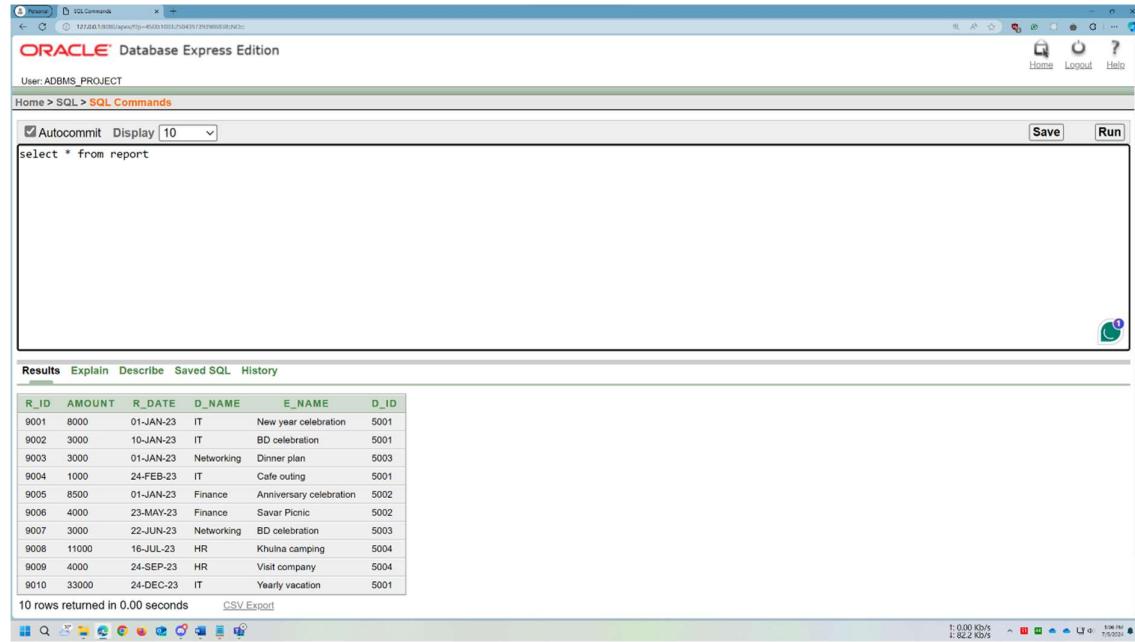
```
INSERT INTO report VALUES (9006, 4000, TO_DATE('2023-05-23', 'YYYY-MM-DD'),'Finance', 'Savar Picnic', 5002);
```

```
INSERT INTO report VALUES (9007, 3000, TO_DATE('2023-06-22', 'YYYY-MM-DD'),'Networking', 'BD celebration', 5003);
```

```
INSERT INTO report VALUES (9008, 11000, TO_DATE('2023-07-16', 'YYYY-MM-DD'),'HR', 'Khulna camping', 5004);
```

```
INSERT INTO report VALUES (9009, 4000, TO_DATE('2023-09-24', 'YYYY-MM-DD'),'HR', 'Visit company', 5004);
```

```
INSERT INTO report VALUES (9010, 33000, TO_DATE('2023-12-24', 'YYYY-MM-DD'),'IT', 'Yearly vacation', 5001);
```



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

```
select * from report
```

The results pane displays the following data:

R_ID	AMOUNT	R_DATE	D_NAME	E_NAME	D_ID
9001	8000	01-JAN-23	IT	New year celebration	5001
9002	3000	10-JAN-23	IT	BD celebration	5001
9003	3000	01-JAN-23	Networking	Dinner plan	5003
9004	1000	24-FEB-23	IT	Cafe outing	5001
9005	8500	01-JAN-23	Finance	Anniversary celebration	5002
9006	4000	23-MAY-23	Finance	Savar Picnic	5002
9007	3000	22-JUN-23	Networking	BD celebration	5003
9008	11000	16-JUL-23	HR	Khulna camping	5004
9009	4000	24-SEP-23	HR	Visit company	5004
9010	33000	24-DEC-23	IT	Yearly vacation	5001

10 rows returned in 0.00 seconds

Query Writing: Basic PL/SQL

Variables:

Q.1 Take user input of employee id and display details.

Query:

```
CREATE OR REPLACE PROCEDURE display_employee_details (
    p_emp_id IN NUMBER) IS
    v_emp_name      VARCHAR2(30);
    v_designation   VARCHAR2(10);
    v_manager_id    NUMBER(4);
BEGIN
    SELECT emp_name, designation, manager_id INTO v_emp_name, v_designation,
        v_manager_id FROM employee WHERE emp_id = p_emp_id;
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || p_emp_id);
    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_emp_name);
    DBMS_OUTPUT.PUT_LINE('Designation: ' || v_designation);
    DBMS_OUTPUT.PUT_LINE('Manager ID: ' || NVL(TO_CHAR(v_manager_id), 'No
Manager'));
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No employee found with ID ' || p_emp_id);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

The screenshot shows the Oracle Application Express SQL Commands interface. In the top navigation bar, there are tabs for Home, SQL, and SQL Commands. The main area contains a code editor with the following PL/SQL code:

```
-- Executing the procedure
BEGIN
    display_employee_details(1004);
END;
/
```

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

Employee ID:	1004
Employee Name:	Chondrima Kumari
Designation:	Developer
Manager ID:	1008
Statement processed.	
0.00 seconds	

At the bottom of the interface, it says "Language: en-gb" and "Copyright © 1999, 2006, Oracle. All rights reserved." The system status bar at the bottom right shows "94°F Haze" and the date "7/7/2024".

Q.2 Take user input of manager's employee id and display employees working under that manager

Query:

```
CREATE OR REPLACE PROCEDURE employees_manager (
    p_manager_id IN NUMBER) IS
    CURSOR emp_cursor IS
        SELECT emp_id, emp_name, designation FROM employee WHERE manager_id =
            p_manager_id;
    v_emp_id    employee.emp_id%TYPE;
    v_emp_name  employee.emp_name%TYPE;
    v_designation employee.designation%TYPE;
    BEGIN
        OPEN emp_cursor;
        LOOP
            FETCH emp_cursor INTO v_emp_id, v_emp_name, v_designation;
            EXIT WHEN emp_cursor%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_emp_id);
        END LOOP;
    END;
```

```

DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_emp_name);
DBMS_OUTPUT.PUT_LINE('Designation: ' || v_designation);
DBMS_OUTPUT.PUT_LINE('-----');
END LOOP;

CLOSE emp_cursor;

EXCEPTION

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.PUT_LINE('No employees found' || p_manager_id);

WHEN OTHERS THEN

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

END;

```

/

```

Employee ID: 1003
Employee Name: Mahmud Al Ashiq
Designation: Engineer
-----
Employee ID: 1006
Employee Name: Rodoshie Reehean
Designation: Engineer
-----
Statement processed.

0.00 seconds

```

Operators:

Q.3 Display the events whose budget is more than 8000Tk.

Query:

```
CREATE OR REPLACE PROCEDURE display_events
IS
CURSOR event_cursor IS
SELECT e_id, e_name, budget, description
FROM event
WHERE budget > 8000;
```

```
v_e_id      event.e_id%TYPE;
v_e_name    event.e_name%TYPE;
v_budget    event.budget%TYPE;
v_description event.description%TYPE;
BEGIN
OPEN event_cursor;
LOOP
FETCH event_cursor INTO v_e_id, v_e_name, v_budget, v_description;
EXIT WHEN event_cursor%NOTFOUND;
```

```
DBMS_OUTPUT.PUT_LINE('Event ID: ' || v_e_id);
DBMS_OUTPUT.PUT_LINE('Event Name: ' || v_e_name);
DBMS_OUTPUT.PUT_LINE('Budget: ' || v_budget || ' Tk');
DBMS_OUTPUT.PUT_LINE('Description: ' || v_description);
DBMS_OUTPUT.PUT_LINE('-----');
END LOOP;
CLOSE event_cursor;
EXCEPTION
WHEN NO_DATA_FOUND THEN
```

```

DBMS_OUTPUT.PUT_LINE('No events found with budget greater than 8000 Tk');

WHEN OTHERS THEN

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

END;

/

```

```

User SCOTT
Home > SQL > SQL Commands
Autocommit Display | 10 | Save Run
BEGIN
    display_events;
END;
-----
```

Results Explain Describe Saved SQL History

```

Event ID: 6005
Event Name: Anniversary celebration
Budget: 8500 Tk
Description: Anniversary of the company celebrated
-----
Event ID: 6008
Event Name: Camping to Khulna
Budget: 11000 Tk
Description: Entire day tent camping held on Khulna
-----
Event ID: 6010
Event Name: Yearly vacation
Budget: 33000 Tk
Description: Vacation was in Sajek for 5 days
-----
```

94°F Haze 3:26 PM 7/7/2024

Q.4 Display the expenses that are less than 6000Tk.

Query:

```

CREATE OR REPLACE PROCEDURE display_expenses IS
CURSOR expense_cursor IS
SELECT exp_id, description, amount, expense_date
FROM expense
WHERE amount < 6000;
v_exp_id      expense.exp_id%TYPE;
v_description expense.description%TYPE;
v_amount      expense.amount%TYPE;
v_expense_date expense.expense_date%TYPE;
BEGIN

```

```

OPEN expense_cursor;
LOOP
  FETCH expense_cursor INTO v_exp_id, v_description, v_amount, v_expense_date;
  EXIT WHEN expense_cursor%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE('Expense ID: ' || v_exp_id);
  DBMS_OUTPUT.PUT_LINE('Description: ' || v_description);
  DBMS_OUTPUT.PUT_LINE('Amount: ' || v_amount || ' Tk');
  DBMS_OUTPUT.PUT_LINE('Expense Date: ' || TO_CHAR(v_expense_date, 'YYYY-MM-DD'));
  DBMS_OUTPUT.PUT_LINE('-----');
END LOOP;
CLOSE expense_cursor;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No expenses found with amount less than 6000 Tk');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;

```

```

SQL Commands
127.0.0.1:8080/apex/f?p=4500:1003:3250382272987313::NO:::
Autocommit Display 10
BEGIN
  display_expenses ;
END;

Results Explain Describe Saved SQL History

Expense ID: 3002
Description: Birthday Surprise to Sumaiya
Amount: 3000 Tk
Expense Date: 2024-01-10
-----
Expense ID: 3004
Description: Sudden Lunch plan to nearby cafe
Amount: 1000 Tk
Expense Date: 2024-02-20
-----
Expense ID: 3005
Description: Picnic to Savar
Amount: 4000 Tk
Expense Date: 2024-05-23
-----
Expense ID: 3007
Description: Birthday Surprise to Lance Florida
Amount: 3000 Tk
Expense Date: 2024-06-22
-----
Expense ID: 3009
Description: Tour to Mindset IT Industry for visit and lunch
Amount: 4000 Tk
Expense Date: 2024-09-11
-----
```

Single-row functions:

Q.5 Display the events between three months

Query:

```
CREATE OR REPLACE PROCEDURE display_events (
    p_start_date IN DATE,
    p_end_date IN DATE) IS
    CURSOR event_cursor IS
        SELECT e_id, e_name, event_date, budget, description
        FROM event
        WHERE event_date BETWEEN p_start_date AND p_end_date;
    v_e_id      event.e_id%TYPE;
    v_e_name    event.e_name%TYPE;
    v_event_date event.event_date%TYPE;
    v_budget    event.budget%TYPE;
    v_description event.description%TYPE;
BEGIN
    OPEN event_cursor;
    LOOP
        FETCH event_cursor INTO v_e_id, v_e_name, v_event_date, v_budget, v_description;
        EXIT WHEN event_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Event ID: ' || v_e_id);
        DBMS_OUTPUT.PUT_LINE('Event Name: ' || v_e_name);
        DBMS_OUTPUT.PUT_LINE('Event Date: ' || TO_CHAR(v_event_date, 'YYYY-MM-DD'));
        DBMS_OUTPUT.PUT_LINE('Budget: ' || v_budget || ' Tk');
        DBMS_OUTPUT.PUT_LINE('Description: ' || v_description);
        DBMS_OUTPUT.PUT_LINE('-----');
    END LOOP;
    CLOSE event_cursor;
```

EXCEPTION

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.PUT_LINE('No events found within the specified date range');

WHEN OTHERS THEN

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

END;

/

```
SQL Commands
127.0.0.1:8060/apex/f?p=4500:1003:3250382272987313::NO::
ORACLE Database Express Edition
User SCOTT
Home > SQL > SQL Commands
Autocommit Display | 10 | Save Run
BEGIN
    display_events(TO_DATE('2024-07-01', 'YYYY-MM-DD'), TO_DATE('2024-10-31', 'YYYY-MM-DD'));
END;

Results Explain Describe Saved SQL History
Event ID: 6008
Event Name: Camping to Khulna
Event Date: 2024-07-16
Budget: 11000 Tk
Description: Entire day tent camping held on Khulna
-----
Event ID: 6009
Event Name: Sister concern IT Firm visited
Event Date: 2024-09-24
Budget: 4000 Tk
Description: Visited Mindset It firm with complementary lunch
-----
```

Q.6 Display total expenses of a single month

Query:

DECLARE

v_month VARCHAR2(7) := '2024-01'; -- Specify the month in 'YYYY-MM' format

v_total_expenses NUMBER;

BEGIN

SELECT SUM(amount)

INTO v_total_expenses

FROM expense

WHERE TO_CHAR(expense_date, 'YYYY-MM') = v_month;

```

DBMS_OUTPUT.PUT_LINE('Total expenses for ' || v_month || ':' || v_total_expenses || '
Tk');

EXCEPTION

WHEN NO_DATA_FOUND THEN

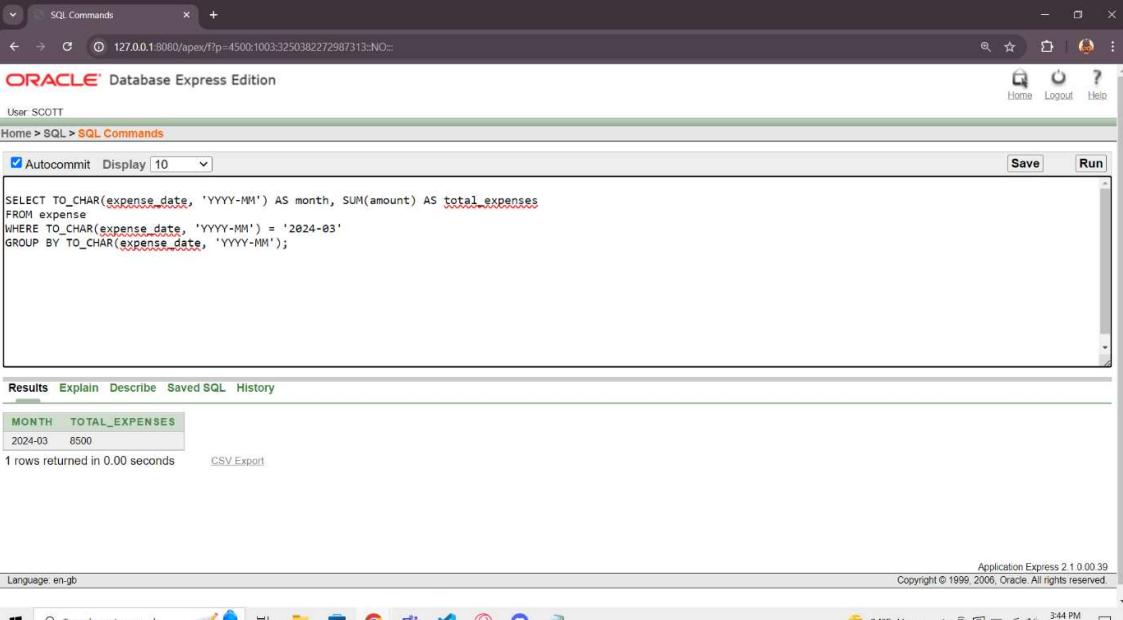
DBMS_OUTPUT.PUT_LINE('No expenses found for ' || v_month);

WHEN OTHERS THEN

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

END;

```



The screenshot shows a SQL command being run in the Oracle Database Express Edition interface. The command is:

```

SELECT TO_CHAR(expense_date, 'YYYY-MM') AS month, SUM(amount) AS total_expenses
FROM expense
WHERE TO_CHAR(expense_date, 'YYYY-MM') = '2024-03'
GROUP BY TO_CHAR(expense_date, 'YYYY-MM');

```

The results table shows:

MONTH	TOTAL_EXPENSES
2024-03	8500

1 rows returned in 0.00 seconds

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

Group function:

Q.7 Count the number of employees managed by each manager.

```

SELECT m.emp_name AS manager_name, COUNT(e.emp_id) AS employee_count
FROM manager m
JOIN employee e ON m.emp_id = e.manager_id
GROUP BY m.emp_name;

```

```

SELECT m.emp_name AS manager_name, COUNT(e.emp_id) AS employee_count
FROM manager m
JOIN employee e ON m.emp_id = e.manager_id
GROUP BY m.emp_name;

```

Results

MANAGER_NAME	EMPLOYEE_COUNT
Baria Ifeen	2
Jessia Islam	2
Tahib Alam	1
Lance Florida	1

4 rows returned in 0.00 seconds

Q.8 Display Total Expenses Grouped by each Event

```

SELECT ev.e_id, ev.e_name, SUM(e.amount) AS total_expense
FROM event ev
JOIN expense e ON ev.exp_id = e.exp_id
GROUP BY ev.e_id, ev.e_name;

```

```

SELECT ev.e_id, ev.e_name, SUM(e.amount) AS total_expense
FROM event ev
JOIN expense e ON ev.exp_id = e.exp_id
GROUP BY ev.e_id, ev.e_name;

```

Results

E_ID	E_NAME	TOTAL_EXPENSE
6003	Dinner Plan	6500
6009	Sister concern IT Firm visited	4000
6008	Camping to Khulna	11000
6006	Picnic	4000
6010	Yearly vacation	33000
6001	New Year Party	8000
6002	Birthday Party	3000
6005	Anniversary celebration	8500
6004	Cafe outing	1000
6007	Birthday Party	3000

10 rows returned in 0.00 seconds

Loop:

Q.9 Display the allocated fund of a department with an additional 5%, 10%, 15%, 20%, 25% increment.

Query:

```
DECLARE
```

```
    a_fund number;
```

```
    increment number:=5;
```

```
BEGIN
```

```
    SELECT a_fund INTO a_fund FROM fund WHERE f_id=4001;
```

```
    dbms_output.put_line('Allocated Fund: '|| a_fund);
```

```
    WHILE increment<30 LOOP
```

```
        dbms_output.put_line(increment||'% increment:'||(a_fund+a_fund*increment/100));
```

```
        increment:= increment+5;
```

```
    END LOOP;
```

```
END;
```

```
/
```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The code entered is a PL/SQL block that retrieves the allocated fund for department ID 4001 and then loops 30 times to calculate and print the fund with 5% increments. The output window displays the initial fund and the results of each iteration of the loop.

```
ORACLE Database Express Edition
User: ADBMS_PROJECT
Home > SQL > SQL Commands
Autocommit: On | Display: 10 | Save | Run | G |
```

```
DECLARE
    a_fund number;
    new_fund number;
    increment number:=5;
BEGIN
    SELECT a_fund INTO a_fund FROM fund WHERE f_id=4001;
    dbms_output.put_line('Allocated Fund: '|| a_fund);
    WHILE increment<30 LOOP
        dbms_output.put_line(increment||'% increment:'||(a_fund+a_fund*increment/100));
        increment:= increment+5;
    END LOOP;
END;
/
```

Results Explain Describe Saved SQL History

```
Allocated Fund: 15000
5% increment:15750
10% increment:16500
15% increment:17250
20% increment:18000
25% increment:18750

Statement processed.
```

Q.10 Display all the managers using loop

Query:

```
DECLARE
id NUMBER;
name VARCHAR2(20);
department VARCHAR2(10);
CURSOR manager IS
SELECT emp_id, emp_name, department FROM manager;
BEGIN
OPEN manager;
LOOP
FETCH manager INTO id, name, department;
EXIT WHEN manager%NOTFOUND;
dbms_output.put_line('ID:'||id||' Name:'||name||' Department:'||department);
END LOOP;
CLOSE manager;
END;
```

/

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

```
DECLARE
id NUMBER;
name VARCHAR2(20);
department VARCHAR2(10);
CURSOR manager IS
SELECT emp_id, emp_name, department FROM manager;
BEGIN
OPEN manager;
LOOP
FETCH manager INTO id, name, department;
EXIT WHEN manager%NOTFOUND;
dbms_output.put_line('ID:'||id||' Name:'||name||' Department:'||department);
END LOOP;
CLOSE manager;
END;
```

The results pane displays the output of the dbms_output.put_line statements:

```
ID:1007 Name:Baria Ifteen Department:IT
ID:1008 Name:Tahsib Alam Department:Networking
ID:1009 Name:Lance Florida Department:Finance
ID:1010 Name:Jessia Islam Department:HR
```

Below the results, it says "Statement processed." and "0.00 seconds". At the bottom right, there is copyright information: "Application Express 2.1.0.0.39" and "Copyright © 1999, 2006, Oracle. All rights reserved."

Conditional statements:

Q.11 Display the event names and their budgets. If the budget is greater than 5000, display 'High Budget Event', otherwise display 'Low Budget Event'.

```
SELECT e_name,
       budget,
       CASE
           WHEN budget > 5000 THEN 'High Budget Event'
           ELSE 'Low Budget Event'
       END AS budget_category
  FROM event;
```

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

```
SELECT e_name,
       budget,
       CASE
           WHEN budget > 5000 THEN 'High Budget Event'
           ELSE 'Low Budget Event'
       END AS budget_category
  FROM event;
```

The Results tab displays the output of the query:

E_NAME	BUDGET	BUDGET_CATEGORY
New Year Party	8000	High Budget Event
Birthday Party	3000	Low Budget Event
Cafe outing	1000	Low Budget Event
Anniversary celebration	8500	High Budget Event
Picnic	4000	Low Budget Event
Birthday Party	3000	Low Budget Event
Camping to Khadha	11000	High Budget Event
Sales team IT Farm visited	2000	Low Budget Event
Yearly vacation	33000	High Budget Event
Dinner Plan	3000	Low Budget Event

At the bottom of the interface, the Windows taskbar is visible, showing various application icons.

Q.12 Display the name and designation of employees. If an employee's designation is 'Manager', display 'Senior Staff', otherwise display 'Junior Staff'.

```
SELECT emp_name,
       designation,
       CASE
           WHEN designation = 'Manager' THEN 'Senior Staff'
           ELSE 'Junior Staff'
       END AS staff_level
  FROM employee;
```

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

```

SELECT emp_name,
       designation,
       CASE
           WHEN designation = 'Manager' THEN 'Senior Staff'
           ELSE 'Junior Staff'
       END AS staff_level
FROM employee;

```

The results window displays the following data:

EMP_NAME	DESIGNATION	STAFF_LEVEL
Sumaya Taslim	Developer	Junior Staff
Nishat Afia	Developer	Junior Staff
Mahmud Al Ashiq	Engineer	Junior Staff
Chondrima Kumar	Developer	Junior Staff
Kamal Hasib	Analyst	Junior Staff
Rodoshie Reheem	Engineer	Junior Staff
Baria Isees	Manager	Senior Staff
Tahsib Alam	Manager	Senior Staff
Lance Florida	Manager	Senior Staff
Jessia Islam	Manager	Senior Staff

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

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

Subquery:

Q.13 Display the employees under the manager of the ‘Networking’ department

Query:

DECLARE

emp_id number;

emp_name varchar2(30);

designation varchar2(10);

BEGIN

SELECT emp_id, emp_name, designation INTO emp_id, emp_name, designation FROM employee WHERE manager_id = (SELECT emp_id FROM manager

WHERE department = 'Networking');

dbms_output.put_line('ID: ' || emp_id);

dbms_output.put_line('Name: ' || emp_name);

dbms_output.put_line('Designation: ' || designation);

END;

/

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

```

DECLARE
    emp_id number;
    emp_name varchar2(30);
    designation varchar2(10);
BEGIN
    SELECT emp_id, emp_name, designation INTO emp_id, emp_name, designation FROM employee WHERE manager_id = (SELECT emp_id FROM manager
    WHERE department = 'Networking');
    dbms_output.put_line('ID: ' || emp_id);
    dbms_output.put_line('Name: ' || emp_name );
    dbms_output.put_line('Designation: ' || designation );
END;
/

```

The results section displays the output of the query:

```

ID: 1004
Name: Chondrima Kumari
Designation: Developer
Statement processed.

0.00 seconds

```

Q.14 Display the alphabetically first sponsor's details

Query:

DECLARE

id number;

name varchar2(30);

description varchar2(100);

BEGIN

SELECT sp_id, description, sp_name INTO id, description, name FROM (SELECT * FROM sponsor ORDER BY sp_name) WHERE ROWNUM = 1;

dbms_output.put_line('ID: ' || id);

dbms_output.put_line('Name: ' || name);

dbms_output.put_line('Designation: ' || description);

END;

/

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

```

DECLARE
  id number;
  name varchar2(30);
  description varchar2(100);
BEGIN
  SELECT sp_id, description, sp_name INTO id, description, name FROM (SELECT * FROM sponsor ORDER BY sp_name) WHERE ROWNUM = 1;
  dbms_output.put_line('ID: ' || id);
  dbms_output.put_line('Name: ' || name );
  dbms_output.put_line('Designation: ' || description );
END;
/

```

The results pane displays the output of the query:

```

ID: 7005
Name: Dolby Company
Designation: Company anniversary expenses for gifts and dinner
Statement processed.

0.00 seconds

```

Joining:

Q.15 Display the event in which budget and expense are the same

Query:

```

DECLARE
  name varchar2(30);
  event_date varchar2(15);
  budget number;
  amount number;
BEGIN
  SELECT event.e_name, event.event_date, event.budget, expense.amount
  INTO name, event_date, budget, amount
  FROM event event, expense expense
  WHERE event.budget = expense.amount AND event.exp_id = expense.exp_id AND
  event.event_date LIKE '%FEB%';
  dbms_output.put_line('Name: ' || name || ' Date: ' || event_date);
  dbms_output.put_line('Budget: ' || budget|| ' Amount: ' || amount);
END;
/

```

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

```

DECLARE
    name varchar2(30);
    event_date varchar2(15);
    budget number;
    amount number;
BEGIN
    SELECT event.e_name, event.event_date, event.budget, expense.amount
    INTO name, event_date, budget, amount
    FROM event event, expense expense
    WHERE event.budget = expense.amount AND event.exp_id = expense.exp_id AND event.event_date LIKE '%FEB%';
    dbms_output.put_line('Name: ' || name || ' Date: ' || event_date);
    dbms_output.put_line('Budget: ' || budget|| ' Amount: ' || amount);
END;
/

```

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

```

Name: Cafe outing      Date: 24-FEB-24
Budget: 1000      Amount: 1000
Statement processed.

0.00 seconds

```

Q.16 Display employee and their manager's information where employee name has 'Ashiq'

Query:

DECLARE

id number;

name varchar2(30);

designation varchar2(15);

mgr_id number;

mgr_name varchar2(30);

department varchar2(15);

BEGIN

SELECT emp.emp_id, emp.emp_name, emp.designation, mgr.emp_id, mgr.emp_name,
mgr.department

INTO id, name, designation, mgr_id, mgr_name, department

FROM employee emp, manager mgr

WHERE emp.manager_id (+) = mgr.emp_id AND emp.emp_name LIKE '%Ashiq%';

dbms_output.put_line('ID: ' || id || ' Name: ' || name || ' Designation: ' || designation);

dbms_output.put_line('Manager ID: ' || mgr_id || ' Name: ' || mgr_name || ' Department: ' || designation);

END;

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

```

DECLARE
  id number;
  name varchar2(30);
  designation varchar2(15);
  mgr_id number;
  mgr_name varchar2(30);
  department varchar2(15);
BEGIN
  SELECT emp.emp_id, emp.emp_name, emp.designation, mgr.emp_id, mgr.emp_name, mgr.department
  INTO id, name, designation, mgr_id, mgr_name, department
  FROM employee emp, manager mgr
  WHERE emp.manager_id (+) = mgr.emp_id AND emp.emp_name LIKE '%Ashiq%';
  dbms_output.put_line('ID: ' || id || ' Name: ' || name || ' Designation: ' || designation);
  dbms_output.put_line('Manager ID: ' || mgr_id || ' Name: ' || mgr_name || ' Department: ' || designation);
END;
/

```

The results section displays the output of the query:

```

ID: 1003  Name: Mahmud Al Ashiq  Designation: Engineer
Manager ID: 1010  Name: Jessia Islam  Department: Engineer
Statement processed.

```

Advance PL/SQL

Stored Function:

Q.17 Create a stored function get_event_budget that takes an event ID as input and returns the budget of that event.

Query:

```

CREATE OR REPLACE FUNCTION get_event_budget (
  p_event_id IN NUMBER
) RETURN NUMBER
IS
  v_budget NUMBER;
BEGIN
  SELECT budget INTO v_budget FROM event WHERE e_id = p_event_id;
  RETURN v_budget;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN NULL;
  WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(-20001, 'An error occurred: ' || SQLERRM);
END;

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. A user named SCOTT is logged in. The query entered is:

```
SELECT get_event_budget(6005) AS budget FROM dual;
```

The results show a single row with a column labeled 'BUDGET' containing the value 8500. The interface includes tabs for Results, Explain, Describe, Saved SQL, and History.



Q.18 Create a stored function `get_employee_designation` that takes an employee ID as input and returns the designation of that employee.

Query:

```
CREATE OR REPLACE FUNCTION get_employee_designation (
    p_emp_id IN NUMBER
) RETURN VARCHAR2
IS
    v_designation VARCHAR2(10);
BEGIN
    SELECT designation INTO v_designation FROM employee WHERE emp_id = p_emp_id;
    RETURN v_designation;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20001, 'An error occurred: ' || SQLERRM);
END;
/
```

User: SCOTT

Home > SQL > SQL Commands

Autocommit

```
SELECT get_employee_designation(1006) AS designation FROM dual;
```

DESIGNATION

Engineer

1 rows returned in 0.00 seconds



Stored Procedure:

Q.19 Create a stored procedure update_event_budget that takes an event ID and a new budget amount as input, then updates the budget for that event.

Query:

```
CREATE OR REPLACE PROCEDURE update_event_budget (
    p_event_id IN NUMBER,
    p_new_budget IN NUMBER) IS
BEGIN
    UPDATE event SET budget = p_new_budget WHERE e_id = p_event_id;
    IF SQL%ROWCOUNT = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Event ID not found');
    END IF;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Budget updated successfully for event ID ' || p_event_id);
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE_APPLICATION_ERROR(-20001, 'An error occurred: ' || SQLERRM);
END;
```

END;

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

```
BEGIN
    update event_budget(6001, 9000); -- Example event ID and new budget
END;
/
select * from event;
```

The results section displays a table with the following data:

E_ID	E_NAME	EVENT_DATE	BUDGET	DESCRIPTION	EXP_ID	MANAGER_ID
6001	New Year Party	01-JAN-24	9000	Picnic done in Kolabagan	3001	1007
6002	Birthday Party	10-JAN-24	3000	Surprise to a IT dept member	3002	1007
6004	Cafe outing	24-FEB-24	1000	Cafe foods supplied	3004	1009
6005	Anniversary celebration	24-MAR-24	8500	Anniversary of the company celebrated	3005	1010
6006	Picnic	10-JAN-24	4000	Savar picnic in early morning	3006	1008
6007	Birthday Party	22-JUN-24	3000	BD Surprise to finance manager	3007	1008
6008	Camping to Khulna	16-JUL-24	11000	Entire day tent camping held on Khulna	3008	1009
6009	Sister concern IT Firm visited	24-SEP-24	4000	Visited Mindset It firm with complementary lunch	3009	1010
6010	Yearly vacation	24-DEC-24	33000	Vacation was in Sajek for 5 days	3010	1010
6003	Dinner Plan	17-FEB-24	3000	Dinner Plan to banani	3003	1007

Q.20 Create a stored procedure add_new_sponsor that takes sponsor details as input and adds a new sponsor to the database.

Query:

```
CREATE OR REPLACE PROCEDURE add_new_sponsor (
    p_sp_id IN NUMBER,
    p_description IN VARCHAR2,
    p_sp_name IN VARCHAR2,
    p_manager_id IN NUMBER) IS
BEGIN
    INSERT INTO sponsor (sp_id, description, sp_name, manager_id)
    VALUES (p_sp_id, p_description, p_sp_name, p_manager_id);
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Sponsor added successfully with ID ' || p_sp_id);
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE_APPLICATION_ERROR(-20001, 'An error occurred: ' || SQLERRM);
END;
/
```

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

```

BEGIN
    add_new_sponsor(7012, 'New Sponsor for fantasy park rides', 'Tech Corp', 1007);
END;
/

```

The results show:

```

Sponsor added successfully with ID 7012
Statement processed.

0.00 seconds

```

At the bottom, it says "Language: en-gb" and "Application Express 2.1.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved."

Table-based Record:

Q.21 Write a table-based record that retrieves a employee details.

DECLARE

employee_record employee%ROWTYPE;

BEGIN

SELECT * INTO employee_record FROM employee WHERE emp_id = 1009;

-- Output the values to check

DBMS_OUTPUT.PUT_LINE('Employee ID: ' || employee_record.emp_id);

DBMS_OUTPUT.PUT_LINE('Employee Name: ' || employee_record.emp_name);

DBMS_OUTPUT.PUT_LINE('Designation: ' || employee_record.designation);

DBMS_OUTPUT.PUT_LINE('Manager ID: ' || employee_record.manager_id);

END;

```

ORACLE Database Express Edition
User ADMIN
Home > SQL > SQL Commands
Autocommit: Display [10]
select * from employee
DECLARE
    employee_record employee%ROWTYPE;
BEGIN
    SELECT * INTO employee_record FROM employee WHERE emp_id = 1009;
    -- Output the values to check
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || employee_record.emp_id);
    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || employee_record.emp_name);
    DBMS_OUTPUT.PUT_LINE('Designation: ' || employee_record.emp_designation);
    DBMS_OUTPUT.PUT_LINE('Manager ID: ' || employee_record.manager_id);
END;

```

Results Explain Describe Saved SQL History

Employee ID: 1009
Employee Name: Lance Florida
Designation: Manager
Manager ID:
Statement processed.
0.03 seconds

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

Q.22 Write a table-based record that retrieves the company's specific fund details.

```

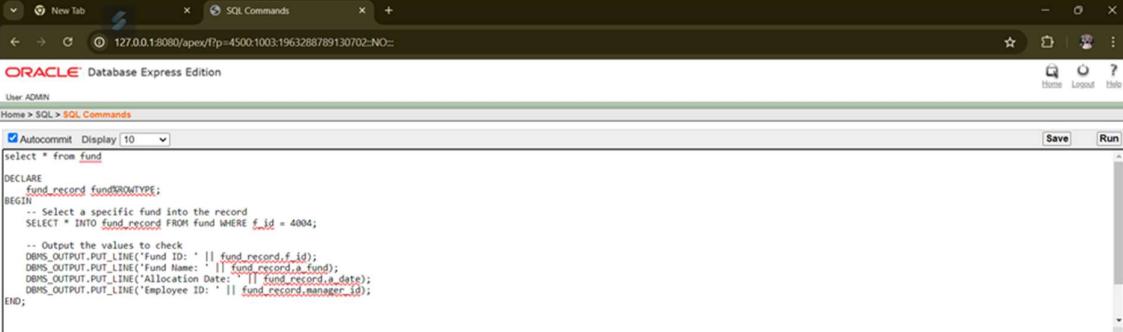
DECLARE
fund_record fund%ROWTYPE;

BEGIN
-- Select a specific fund into the record
SELECT * INTO fund_record FROM fund WHERE f_id = 4004;

-- Output the values to check
DBMS_OUTPUT.PUT_LINE('Fund ID: ' || fund_record.f_id);
DBMS_OUTPUT.PUT_LINE('Fund Name: ' || fund_record.a_fund);
DBMS_OUTPUT.PUT_LINE('Allocation Date: ' || fund_record.a_date);
DBMS_OUTPUT.PUT_LINE('Employee ID: ' || fund_record.manager_id);

END;

```



```

ORACLE Database Express Edition
User ADMIN
Home > SQL > SQL Commands
Autocommit: Display: 10 | Save | Run
select * from fund
DECLARE
    fund_record fund%TYPE;
BEGIN
    -- Select a specific fund into the record
    SELECT * INTO fund_record FROM fund WHERE f_id = 4004;
    -- Output the values to check
    DBMS_OUTPUT.PUT_LINE('Fund ID: ' || fund_record.f_id);
    DBMS_OUTPUT.PUT_LINE('Fund Name: ' || fund_record.f_name);
    DBMS_OUTPUT.PUT_LINE('Allocation Date: ' || fund_record.allocation_date);
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || fund_record.manager_id);
END;

```

Results Explain Describe Saved SQL History

Fund ID: 4004
Fund Name: 9000
Allocation Date: 06-JAN-24
Employee ID: 1010
Statement processed.

0.02 seconds

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

Explicit Cursor:

Q.23 Create an explicit cursor employee_cursor that retrieves all employees working under a specific manager.

DECLARE

```

v_emp_id employee.emp_id%TYPE;
v_emp_name employee.emp_name%TYPE;
v_designation employee.designation%TYPE;
v_manager_id employee.manager_id%TYPE;

```

CURSOR employee_cursor IS

```

SELECT emp_id, emp_name, designation, manager_id
FROM employee
WHERE manager_id = 1007;
BEGIN

```

OPEN employee_cursor;

LOOP

```

FETCH employee_cursor INTO v_emp_id, v_emp_name, v_designation, v_manager_id;
EXIT WHEN employee_cursor%NOTFOUND;

```

```

DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_emp_id);
DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_emp_name);
DBMS_OUTPUT.PUT_LINE('Designation: ' || v_designation);
DBMS_OUTPUT.PUT_LINE('Manager ID: ' || v_manager_id);
END LOOP;
CLOSE employee_cursor;
END;

```

```

User ADMIN
Home > SQL > SQL Commands
Autocommit: Display: 10
select * from employee
DECLARE
    v_emp_id employee.emp_id%TYPE;
    v_emp_name employee.emp_name%TYPE;
    v_designation employee.designation%TYPE;
    v_manager_id employee.manager_id%TYPE;
CURSOR employee_cursor IS
    SELECT emp_id, emp_name, designation, manager_id
    FROM employee
    WHERE manager_id = 1007;
BEGIN
    OPEN employee_cursor;
    LOOP
        FETCH employee_cursor INTO v_emp_id, v_emp_name, v_designation, v_manager_id;
        EXIT WHEN employee_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_emp_id);
        DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_emp_name);
        DBMS_OUTPUT.PUT_LINE('Designation: ' || v_designation);
        DBMS_OUTPUT.PUT_LINE('Manager ID: ' || v_manager_id);
    END LOOP;
    CLOSE employee_cursor;
END;
/

```

Result Explain Describe Saved SQL History

```

Employee ID: 1001
Employee Name: Sumaiya Tasnim
Designation: Developer
Manager ID: 1007
Employee ID: 1002
Employee Name: Aishah Afia
Designation: Developer
Manager ID: 1007

```

Q.24 Create an explicit cursor event_cursor that retrieves all event with a specified manager ID.

```

DECLARE
CURSOR event_cursor IS
SELECT e_id, e_name, event_date, budget, description, exp_id, manager_id
FROM event
WHERE manager_id = 1009;

```

```

v_e_id event.e_id%TYPE;
v_e_name event.e_name%TYPE;
v_event_date event.event_date%TYPE;
v_budget event.budget%TYPE;
v_description event.description%TYPE;

```

```

v_exp_id event.exp_id%TYPE;
v_manager_id event.manager_id%TYPE;

BEGIN
OPEN event_cursor;
LOOP
FETCH event_cursor INTO v_e_id, v_e_name, v_event_date, v_budget, v_description,
v_exp_id, v_manager_id;
EXIT WHEN event_cursor%NOTFOUND;

DBMS_OUTPUT.PUT_LINE('Event ID: ' || v_e_id);
DBMS_OUTPUT.PUT_LINE('Event Name: ' || v_e_name);
DBMS_OUTPUT.PUT_LINE('Event Date: ' || v_event_date);
DBMS_OUTPUT.PUT_LINE('Budget: ' || v_budget);
DBMS_OUTPUT.PUT_LINE('Description: ' || v_description);
DBMS_OUTPUT.PUT_LINE('Expense ID: ' || v_exp_id);
DBMS_OUTPUT.PUT_LINE('Manager ID: ' || v_manager_id);
END LOOP;
CLOSE event_cursor;
END;

```

The screenshot shows the Oracle SQL Developer interface with a SQL Commands tab open. The code in the editor window is a PL/SQL block that retrieves event details from a cursor and prints them using DBMS_OUTPUT.PUT_LINE. The results pane at the bottom shows two rows of output corresponding to the two events in the database.

```

Event ID: 6004
Event Name: Cafe outing
Event Date: 24-FEB-24
Budget: 1000
Description: Cafe foods supplied
Expense ID: 3004
Manager ID: 1009
Event ID: 6009
Event Name: Camping to Khulna
Event Date: 16-JUL-24
Budget: 11000
Description: Entire day tent camping held on Khulna
Expense ID: 3008
Manager ID: 1009

```

Cursor-based Record:**Q.25 Create a cursor-based record that stores HR department's all reports.**

```
DECLARE
CURSOR report_cursor IS
SELECT * FROM report
where d_name='HR';

report_record report%ROWTYPE;
BEGIN
OPEN report_cursor;
LOOP
FETCH report_cursor INTO report_record;
EXIT WHEN report_cursor%NOTFOUND;

DBMS_OUTPUT.PUT_LINE('Report ID: ' || report_record.r_id);
DBMS_OUTPUT.PUT_LINE('Amount: ' || report_record.amount);
DBMS_OUTPUT.PUT_LINE('Report Date: ' || report_record.r_date);
DBMS_OUTPUT.PUT_LINE('Department Name: ' || report_record.d_name);
DBMS_OUTPUT.PUT_LINE('Event Name: ' || report_record.e_name);
DBMS_OUTPUT.PUT_LINE('Department ID: ' || report_record.d_id);
END LOOP;
CLOSE report_cursor;
END;
```

```

ORACLE Database Express Edition
User ADMIN
Home > SQL > SQL Commands
Autocommit: Display: 10 | Save | Run
DECLARE
    CURSOR report_cursor IS
        SELECT * FROM report
        where d_name='HR';
    report_record report%ROWTYPE;
BEGIN
    OPEN report_cursor;
    LOOP
        FETCH report_cursor INTO report_record;
        EXIT WHEN report_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Report ID: ' || report_record.r_id);
        DBMS_OUTPUT.PUT_LINE('Amount: ' || report_record.amount);
        DBMS_OUTPUT.PUT_LINE('Report Date: ' || report_record.r_date);
        DBMS_OUTPUT.PUT_LINE('Department Name: ' || report_record.d_name);
        DBMS_OUTPUT.PUT_LINE('Event Name: ' || report_record.e_name);
        DBMS_OUTPUT.PUT_LINE('Department ID: ' || report_record.d_id);
    END LOOP;
    CLOSE report_cursor;
END;
/

```

Report ID: 9008
Amount: 11000
Report Date: 16-JUL-23
Department Name: HR
Event Name: China camping
Department ID: 5004
Report ID: 9004
Amount: 4000
Report Date: 24-SEP-23
Department Name: HR
Event Name: Visit company
Department ID: 5004

Q.26 Create a cursor-based record to store proposal details by each department.

DECLARE

CURSOR proposal_cursor IS

SELECT * FROM proposal

WHERE d_id = 5002;

proposal_record proposal%ROWTYPE;

BEGIN

OPEN proposal_cursor;

LOOP

FETCH proposal_cursor INTO proposal_record;

EXIT WHEN proposal_cursor%NOTFOUND;

DBMS_OUTPUT.PUT_LINE('Proposal ID: ' || proposal_record.p_id);

DBMS_OUTPUT.PUT_LINE('Description: ' || proposal_record.description);

DBMS_OUTPUT.PUT_LINE('Proposal Date: ' || proposal_record.p_date);

DBMS_OUTPUT.PUT_LINE('Proposal Budget: ' || proposal_record.p_budget);

```

DBMS_OUTPUT.PUT_LINE('Status: ' || proposal_record.status);
DBMS_OUTPUT.PUT_LINE('Employee ID: ' || proposal_record.manager_id);
DBMS_OUTPUT.PUT_LINE('Department ID: ' || proposal_record.d_id);
END LOOP;

CLOSE proposal_cursor;
END;

```

The screenshot shows the Oracle Application Express interface. At the top, there's a browser-like header with tabs for 'New Tab' and 'SQL Commands'. Below it is a code editor window containing a PL/SQL block. The code declares a cursor 'proposal_cursor' that selects from the 'proposal' table where 'd_id' is 5002. It then loops through this cursor, fetching records into a 'proposal_record' rowtype and printing various fields using DBMS_OUTPUT.PUT_LINE. The results tab shows two rows of data:

Proposal ID	Description	Proposal Date	Proposal Budget	Status	Employee ID	Department ID
8005	Company Anniversary	01-MAR-24	8500	completed	1008	5002
8006	Picnic to savor	02-MAY-24	4000	planned	1008	5002

Below the results, a message says 'Statement processed.' and '0.01 seconds'. The bottom part of the screenshot shows the Windows taskbar with various application icons.

Row-level Trigger:

Q.27 Write a row-level trigger log_event_update that logs any updates made to the events table.

Query:

```

CREATE OR REPLACE TRIGGER log_event_update
BEFORE UPDATE ON event
FOR EACH ROW
WHEN (NEW.e_id > 0)
BEGIN
    dbms_output.put_line('Old Budget: '||OLD.budget);
    dbms_output.put_line('Updated Budget: '||NEW.budget);
END;

```

/

UPDATE event SET budget=3300 WHERE e_id=6010;

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

```
CREATE OR REPLACE TRIGGER log_event_update
BEFORE UPDATE ON event
FOR EACH ROW
WHEN (NEW.e_id > 0)
BEGIN
    dbms_output.put_line('Old Budget: ' || :OLD.budget);
    dbms_output.put_line('Updated Budget: ' || :NEW.budget);
END;
/
UPDATE event SET budget=3300 WHERE e_id=6010;
```

The results pane shows the output of the trigger execution:

```
Old Budget: 33000
Updated Budget: 3300
1 row(s) updated.

0.00 seconds
```

At the bottom, the status bar indicates "Application Express 2.1.0.00.39" and "Copyright © 1999, 2006, Oracle. All rights reserved."

Q.28 Write a row-level trigger sponsor_added that logs any new sponsor added to the sponsor's table.

Query:

```
CREATE OR REPLACE TRIGGER sponsor_added
```

```
AFTER INSERT ON sponsor
```

```
FOR EACH ROW
```

```
WHEN (NEW.sp_id > 0)
```

```
BEGIN
```

```
dbms_output.put_line('New Sponsor Added');
```

```
END;
```

```
/
```

```
INSERT INTO sponsor VALUES (7011,'Yearly picnic', 'KSI Corp', 1008);
```

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

```

CREATE OR REPLACE TRIGGER sponsor_added
AFTER INSERT ON sponsor
FOR EACH ROW
WHEN (NEW.sp_id > 0)
BEGIN
  dbms_output.put_line('New Sponsor Added');
END;
/
INSERT INTO sponsor VALUES (7011,'Yearly picnic', 'KSI Corp', 1008);

```

The results pane shows the output of the trigger execution:

```

New Sponsor Added
1 row(s) inserted.

0.02 seconds

```

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

Statement-level Trigger:

Q.29 Write a statement-level trigger log_report_changes that logs whenever any report details are changed.

Query:

```

CREATE OR REPLACE TRIGGER log_report_changes
BEFORE INSERT OR UPDATE OR DELETE ON report
FOR EACH ROW
WHEN (NEW.r_id > 0)
BEGIN
  dbms_output.put_line('Old amount:'||:OLD.amount);
  dbms_output.put_line('Updated amount:'||:NEW.amount);
END;
/
UPDATE report SET amount=3900 WHERE r_id>9008;

```

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

```

CREATE OR REPLACE TRIGGER log_report_changes
BEFORE INSERT OR UPDATE OR DELETE ON report
FOR EACH ROW
WHEN (NEW.r_id > 0)
BEGIN
    dbms_output.put_line('Old amount: '||:OLD.amount);
    dbms_output.put_line('Updated amount: '||:NEW.amount);
END;
/
UPDATE report SET amount=3900 WHERE r_id>9008;

```

The results section shows the output of the trigger execution:

```

Old amount: 4000
Updated amount: 3900
Old amount: 33000
Updated amount: 3900

2 row(s) updated.

0.00 seconds

```

At the bottom right, it says "Application Express 2.1.0.00.39".

Q.30 Write a statement-level trigger log_employee_changes that logs whenever any employee details are changed.

Query:

```

CREATE OR REPLACE TRIGGER log_employee_changes
BEFORE INSERT OR UPDATE OR DELETE ON employee
FOR EACH ROW
WHEN (NEW.emp_id > 0)
BEGIN
    dbms_output.put_line('Old manager id: '||:OLD.manager_id);
    dbms_output.put_line('New manager id: '||:NEW.manager_id);
END;
/
UPDATE employee SET manager_id=1009 WHERE emp_id IN (1002, 1003);

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. In the SQL editor, a trigger named 'log_employee_changes' is defined:

```

CREATE OR REPLACE TRIGGER log_employee_changes
BEFORE INSERT OR UPDATE OR DELETE ON employee
FOR EACH ROW
WHEN (NEW.emp_id > 0)
BEGIN
    dbms_output.put_line('Old manager id: '||:OLD.manager_id);
    dbms_output.put_line('New manager id: '||:NEW.manager_id);
END;
/

```

Below the trigger definition, an UPDATE statement is executed:

```
UPDATE employee SET manager_id=1009 WHERE emp_id IN (1002, 1003);
```

The results of the query show the output of the DBMS_OUTPUT.PUT_LINE statements:

```

Old manager id: 1007
New manager id: 1009
Old manager id: 1010
New manager id: 1009

```

It also indicates that 2 row(s) were updated and the operation took 0.00 seconds.

Package:

Q.31 Create a package event_mgmt that includes the functions, and procedures defined above.

Package creation:

```

CREATE OR REPLACE PACKAGE event_mgmt AS
FUNCTION get_event_budget (p_event_id IN NUMBER) RETURN NUMBER;
PROCEDURE update_event_budget (p_event_id IN NUMBER,p_new_budget IN NUMBER);
END event_mgmt;

```

Package body creation:

```

CREATE OR REPLACE PACKAGE BODY event_mgmt AS
FUNCTION get_event_budget (p_event_id IN NUMBER) RETURN NUMBER AS
v_budget NUMBER;
BEGIN
SELECT budget INTO v_budget FROM event WHERE e_id = p_event_id;
RETURN v_budget;
END;

```

```

PROCEDURE update_event_budget (p_event_id IN NUMBER, p_new_budget IN
NUMBER) AS
BEGIN
UPDATE event SET budget = p_new_budget WHERE e_id = p_event_id;
IF SQL%ROWCOUNT = 0 THEN
RAISE_APPLICATION_ERROR(-20002, 'Event ID not found');
END IF;
COMMIT;
DBMS_OUTPUT.PUT_LINE('Budget updated successfully for event ID ' || p_event_id);
END;
END event_mgmt;

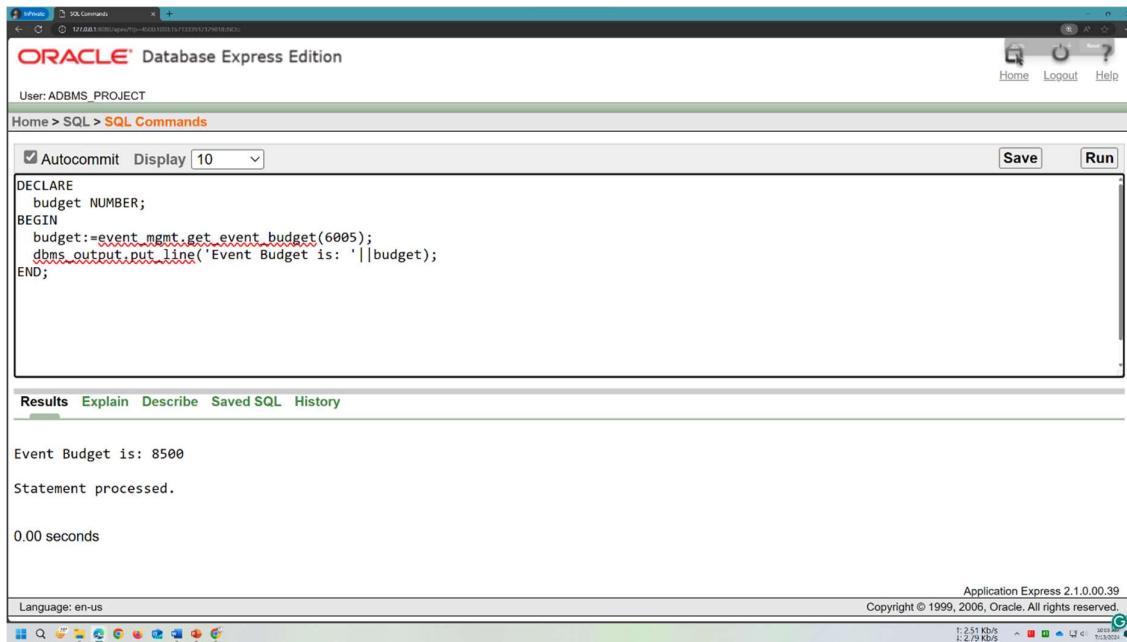
```

Package calling:

```

DECLARE
budget NUMBER;
BEGIN
budget:=event_mgmt.get_event_budget(6005);
dbms_output.put_line('Event Budget is: '||budget);
event_mgmt.update_event_budget(6005, 4500);
END;

```



The screenshot shows the Oracle Database Express Edition Application Express interface. The user is executing a PL/SQL block. The code is as follows:

```

DECLARE
    budget NUMBER;
BEGIN
    budget:=event_mgmt.get_event_budget(6005);
    dbms_output.put_line('Event Budget is: '||budget);
END;

```

The output pane shows the results of the execution:

```

Event Budget is: 8500
Statement processed.

0.00 seconds

```

The bottom status bar indicates the application version is 2.1.0.00.39 and the copyright is from 1999-2006 Oracle.

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

```
BEGIN
    event_mgmt.update_event_budget(6005, 4500);
END;
```

The results section displays the output of the executed command:

```
Old Budget: 8500
Updated Budget: 4500
Budget updated successfully for event ID 6005

Statement processed.
```

Execution time: 0.00 seconds

At the bottom, the Application Express version is shown as 2.1.0.00.39, and the copyright notice is "Copyright © 1999, 2006, Oracle. All rights reserved".

Conclusion:

For future improvements, the system could benefit from enhanced automation features, such as automated fund approval workflows, and better integration of employee feedback through digital surveys. Additionally, introducing predictive analytics could help in better forecasting event budgets and managing resources more efficiently, further optimizing the process for both the management and employees.