DBMS LAB->11
106122112

Q1.a) XML File

(EmployeeDetails.xml)

Copy code

```
<EmployeeDetails>
 <Employee>
 <EmpNo>101</EmpNo>
 <EName>John Smith</EName>
<Job>Manager</Job>
 <WorkingHours>9</WorkingHours>
<Dept>Research</Dept>
 <DeptNo>1</DeptNo>
 <Salary>50000</Salary>
 </Employee>
 <Employee>
 <EmpNo>102</EmpNo>
 <EName>Jane Doe</EName>
<Job>Developer</Job>
 <WorkingHours>8</WorkingHours>
<Dept>Development</Dept>
<DeptNo>2</DeptNo>
 <Salary>40000</Salary>
 </Employee>
 <Employee>
 <EmpNo>103</EmpNo>
 <EName>Mike Johnson</EName>
<Job>Analyst</Job>
 <WorkingHours>7</WorkingHours>
```

<Dept>Research</Dept>

<DeptNo>1</DeptNo>

<Salary>35000</Salary>

</Employee>
<Employee>

<EmpNo>104</EmpNo>

<EName>Susan Lee</EName>

<Job>Tester</Job>

<WorkingHours>10</WorkingHours>

<Dept>Quality Assurance</Dept>

<DeptNo>3</DeptNo>

<Salary>28000</Salary>

</Employee>

<Employee>

<EmpNo>105</EmpNo>

<EName>Robert Smith</EName>

<Job>HR</Job>

<WorkingHours>6</WorkingHours>

<Dept>Human Resources</Dept>

<DeptNo>4</DeptNo>

<Salary>32000</Salary>

</Employee>

</EmployeeDetails>

XQuery Statements

xquery

for $e in /EmployeeDetails/Employee[Salary > 30000]

return $e/Salary

xquery

for $e in /EmployeeDetails/Employee[contains(substring-after(EName, '

'), 'S')]

return $e/EmpNo

```
xquery
for $e in /EmployeeDetails/Employee[Dept =
"Research"] return $e/EName
xquery
for $e in /EmployeeDetails/Employee[WorkingHours >
8] return $e

xquery
for $e in /EmployeeDetails/Employee
order by $e/Salary descending
return $e/Salary

xquery
for $e in /EmployeeDetails/Employee
order by $e/EName
return $e/EName
```

b) XML File (FlightDetails.xml)

xml

```
Copy code
<FlightDetails>
 <Flight>
 <FlNo>101</FlNo>
 <FlName>Alpha Flight</FlName>
 <PilotName>Captain John</PilotName>
<From>New York</From>
 <To>Los Angeles</To>
 <Date>2024-11-01</Date>
 <DepartsTime>10:00</DepartsTime>
```

<ArrivesTime>13:00</ArrivesTime>

<Price>4500</Price>

</Flight>

<Flight>

<FlNo>102</FlNo>
<FlName>Beta Flight</FlName>

<PilotName>Captain Jane</PilotName>

<From>Chicago</From>

<To>Miami</To>

<Date>2024-11-02</Date>

<DepartsTime>14:00</DepartsTime>

<ArrivesTime>17:00</ArrivesTime>

<Price>6000</Price>

</Flight>

<Flight>

<FlNo>103</FlNo>

<FlName>Gamma Flight</FlName>

<PilotName>Captain Mike</PilotName>

<From>San Francisco</From> <To>New

York</To>

<Date>2024-11-01</Date>

<DepartsTime>16:00</DepartsTime>

<ArrivesTime>23:00</ArrivesTime>

<Price>3000</Price>

</Flight>

<Flight>

<FlNo>104</FlNo>

<FlName>Delta Flight</FlName>

<PilotName>Captain Sarah</PilotName>

<From>Los Angeles</From>

<To>Chicago</To>

<Date>2024-11-03</Date>

<DepartsTime>09:00</DepartsTime>

<ArrivesTime>12:00</ArrivesTime>

<Price>4000</Price>

</Flight>

<Flight>
<FlNo>105</FlNo>

<FlName>Epsilon Flight</FlName>

<PilotName>Captain Steve</PilotName>

<From>Miami</From>

<To>San Francisco</To>

<Date>2024-11-01</Date>

<DepartsTime>11:00</DepartsTime>

<ArrivesTime>18:00</ArrivesTime>

<Price>5500</Price>

</Flight>

</FlightDetails>

XQuery Statements

List the price of journeys < 5000:

xquery

Copy code

```
for $f in /FlightDetails/Flight[Price < 5000]
return $f/Price
```

Find the departing time of a particular flight on a particular date

from a particular city:

xquery

```
let $flightNo := '101'
let $date := '2024-11-01'
let $fromCity := 'New York'
```

return

/FlightDetails/Flight[FlNo = $flightNo and Date = $date and From =

$fromCity]/DepartsTime

Find the flight names handled by a particular pilot:

xquery

let $pilotName := 'Captain John'
return

/FlightDetails/Flight[PilotName = $pilotName]/FlName

xquery

let $flightNo := '101'

let $date := '2024-11-01'

return

count(/FlightDetails/Flight[FlNo = $flightNo and Date = $date])

xquery

let $flightNo := '103'

let $date := '2024-11-01'

let $fromCity := 'San Francisco'

return

/FlightDetails/Flight[FlNo = $flightNo and Date = $date and From

= $fromCity]/ArrivesTime

Q2) a. Display details of an employee by ID:

```
CREATE PROCEDURE GetEmployeeDetails(emp_id INT)
BEGIN
 SELECT * FROM Employee WHERE EmpNo = emp_id;
END;
```

b. Add a new employee:

```
CREATE PROCEDURE AddEmployee(emp_no INT, emp_name VARCHAR(50), job
VARCHAR(50), dept VARCHAR(50), dept_no INT, salary INT)
BEGIN
 INSERT INTO Employee (EmpNo, EName, Job, Dept, DeptNo, Salary)
VALUES (emp_no, emp_name, job, dept, dept_no, salary);
END;
```

c. Increase salary:

```
CREATE PROCEDURE RaiseSalary(emp_id INT, hike_amount INT)
BEGIN
 UPDATE Employee SET Salary = Salary + hike_amount WHERE EmpNo =
emp_id;
END;
```

d. Delete employee record by name:

```
CREATE PROCEDURE DeleteEmployee(emp_name
VARCHAR(50)) BEGIN
 DELETE FROM Employee WHERE EName = emp_name;
END;
```

e. List employees in a department:

```
CREATE PROCEDURE ListEmployeesByDept(dept_no INT)
BEGIN
 SELECT EName FROM Employee WHERE DeptNo = dept_no;
END;
```

f. List highest salary in each department:

```
CREATE PROCEDURE ListDeptHighestSalary()
BEGIN
 SELECT DeptNo, MAX(Salary) AS HighestSalary FROM Employee GROUP BY
DeptNo;
END;
```

g. Function for minimum salary:

```
CREATE FUNCTION MinSalary() RETURNS INT
BEGIN
 RETURN (SELECT MIN(Salary) FROM Employee);
END;
```

Q3) -- 1. Disable autocommit for the current session
SET autocommit = 0;

-- 2. Start a new transaction
START TRANSACTION.

-- 3. Update an attribute in the Employee table (for example, Salary of employee with EmpNo
UPDATE Employee SET Salary = Salary + 5000 WHERE EmpNo = 1;

-- 4. Set a savepoint after this update
SAVEPOINT AfterSalaryUpdate.

-- 5. Make another change (e.g., updating Job title)
UPDATE Employee SET Job = 'Senior Manager' WHERE EmpNo = 1;

-- 6. Roll back to the previous savepoint, so the Job title change is undone, but the salary update remains
ROLLBACK TO AfterSalaryUpdate.

-- 7. Commit the transaction, making the salary update permanent
COMMIT;

-- 8. Re-enable autocommit mode if needed
SET autocommit = 1;