Tutorial upload as part of the assessment.

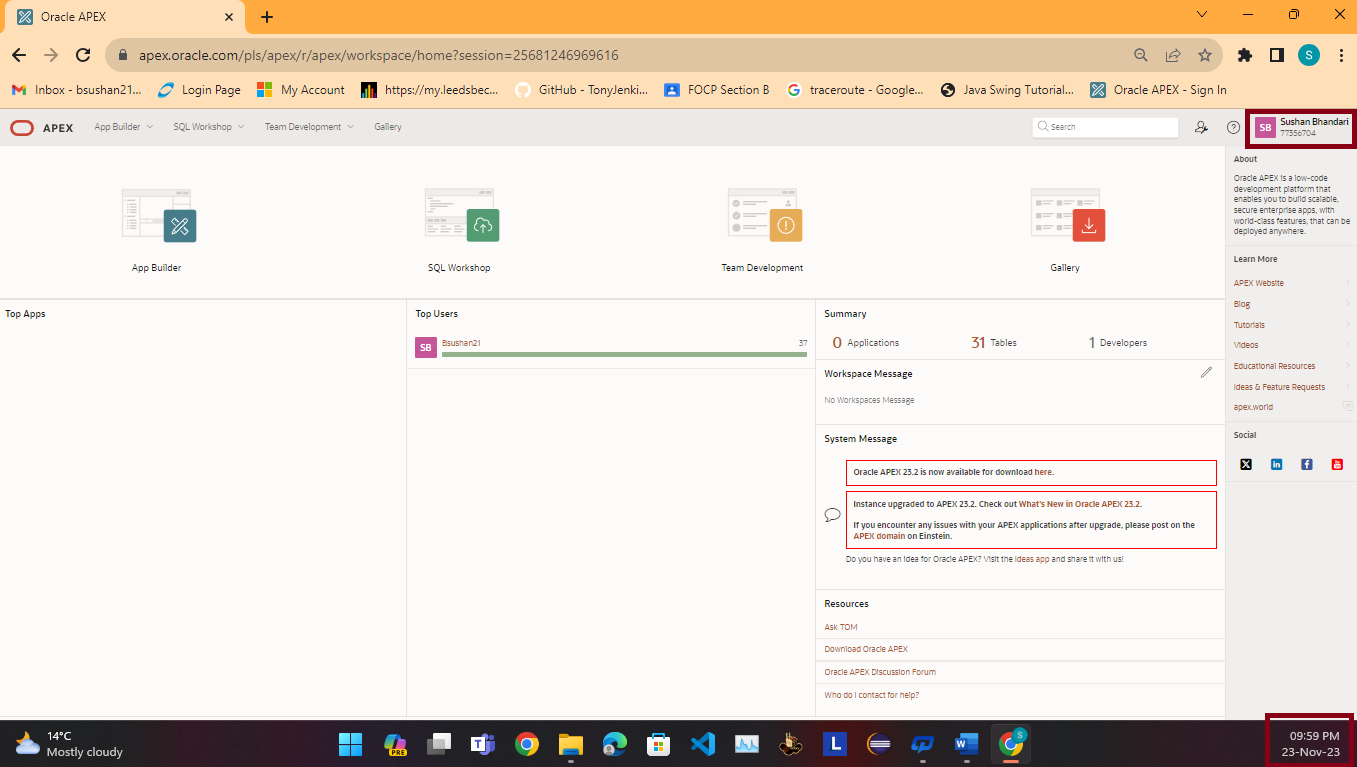
Student Name: Sushan Bhandari

Tutorial Tasks Assignment Title: Design/Implementation

Note: Submit evidence showing the process and workings for your weekly tutorial Apex tasks.

Take screenshots showing clearly:

1. Student id in Apex account
2. Apex 5 – university www address
3. Time and date on your machine

E.g. 

Important note: We expect to see all Recent activities in the Apex account. If these are blank or older then expected during the semester run (e.g. no activity or no activity for a few weeks) we will assume that your code is result of poor practice, as it is impossible to write a fully functional code without a practice.

E.g. Screenshot of SQL Workshop Home page (include as many as you need to evidence your 12 weeks engagement).

A screenshot of a computer

Description automatically generated

Screenshot of SQL scripts environment (include as many as you need to evidence your 12 weeks engagement)

Screenshot of SQL Commands – SAVED scripts (include as many as you need to evidence your 12 weeks engagement) – this is optional, if not used

A screenshot of a computer

Description automatically generated

**Tasks: Here submit evidence of completing the weekly tutorial as per the instructions in the Module Handbook – Assessment section.**

Note that here you will show screenshots with same key info as above, just evidence that tasks have been completed and process of how it was done, as per specified in the task.

You can copy the task here from the Module handbook and provide your solution below, for each task.

**Task 1 – Extended Entity-Relationship Model**

**Holiday Package**

Each holiday package is provided for a certain number of customers. Each holiday package is categorised as being of a specific type e.g. Beach, Winter Sun, Skiing and Adventure, each of which have specific, unique requirements. Each package will fall into one of two journey types, either

* Return flight to a single destination
* Return coach transport with two destinations.

Holidays may have a courier, although some do not.

Where group bookings are made the company needs to record information about the member of the group who is the Group leader; i.e. the arranger of the holiday.

Step 1: Underline all nouns in the text above and list them below:

1. **Holiday Package**
2. **Customers**
3. **Type**
4. Beach
5. Winter Sun
6. Skiing
7. Adventure
8. Requirement
9. Journey
10. **Flight**
11. **Coach**
12. **Destination**
13. **Courier**
14. Company
15. **Group Leader**
16. Member
17. Arranger

Step 2: Ask yourself - Is your noun

* an attribute of an entity, or
* an occurrence of an attribute or
* an entity, which would have more than ONE occurrences and is it relevant to our system (case study)?

Step 3: For each entity create a table to defining attributes and occurrences.

Holiday Package

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute’s Name | **Package\_id** | **Name** | **Duration** | **Cost** | **Accommodation** |
| Occurrences | 1 | Winter Cabin | 3 Days | 20000 | Sushan resort |
| Occurrences | 2 | wildlife | 2 days | 12340 | Balkhu hotel |

**Type**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute’s Name | **Type\_id** | **Name** | **Requirements** |
| Occurrences | 1 | Beach | Life Jacket |
| Occurrences | 2 | Winter Sun | Heater |

**Journey**

|  |  |  |
| --- | --- | --- |
| Attribute’s Name | Journey\_Id | Type |
| Occurrences | 1 | Null |
| Occurrences | 2 | null |

Group Bookings

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attribute’s Name | **Group\_id** | **Group Leader** | **Contact** | **Email** | **Members** |
| Occurrences | 1 | Sushan | 9876567654 | [Sushan@gmail.com](mailto:Sushan@gmail.com) | 6 |
| Occurrences | 2 | Fucheya | 9876789786 | [F@gmail.com](mailto:F@gmail.com) | 7 |

Company

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute’s Name | Company\_Id | Name | Address |
| Occurrences | 1 | Cj tours | Naikap |
| Occurrences | 2 | Xj tarvels | Kalanki |

Member

|  |  |  |
| --- | --- | --- |
| Attribute’s Name | Member\_No | Type |
| Occurrences | 1 | Monthly |
| Occurrences | 2 | Quater |

Step 4: Draw your ERD, by defining entities, relationships, relationship names cardinality.

A diagram of a group

Description automatically generated

Task 2 (got A, B, C activities) - Logical Design and Normalisation

**A.**

Fill in the missing Entity names.

**Customer**(Custid,

**Order** (Orderid, ….*.Custid*

A black line on a white background

Description automatically generated

**Order** (Orderid,

**Item** (Itemid,…*Order\_id*

A black line on a white background

Description automatically generated

**Order** (Orderid,

**Product** (Product\_id

**Item** (Itemid,…***Order\_id, Product\_id***

A white rectangular object with black text

Description automatically generated

1. In this task you are asked to produce an ERD using the given relations and the keys (logical design in reverse).

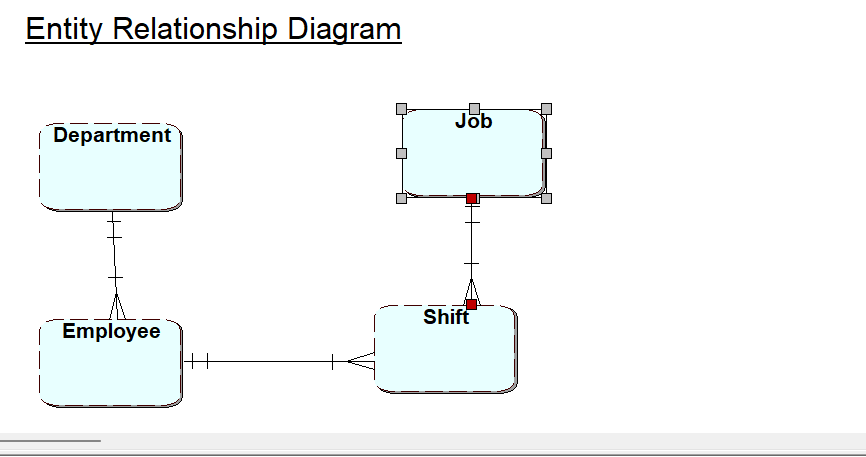
**Department** (Dept\_id, …)

**Employee** (Emp\_id,….., *Dept\_id)*

**Job** (Job\_id,….)

**Shift (***Job\_id, Emp\_id,….)*

****

****

**B.**

Derive this model fully by defining ***foreign keys*** for the relevant tables.

Below is the ERD’s description:

*Every project requires a number of employees. It is usual for consultants to work on more than one project at a time. Each project is broken down into individual tasks which are allocated to specific consultants. Some tasks are common to any project. Consultants meet regularly with clients by making appointments with them.*

Note: When you are completing the task to derive keys from your ERD, no retrospective changes should happen to the ERD design.

Diagram

Description automatically generated

**Derived Tables following Relational Data Model(RDM) rules**

**Step 1: Identify Relations/tables and attributes**

**Step 2: Identify Primary key(s) for each relation/table**

**Step 3: Derive Foreign Keys following the RDM rules**

**Steps 1 and 2 have already been completed for you below.**

Step 3:

Task(Task\_id, Task\_Name,…

Project( Project\_id, Project\_StartDate, Project\_EndDate,…

Project\_Task(PT\_id, Desc,…

Consultant(Consultant\_id, Name, Address, Phone,…

Client (Client\_id, Name, Address, Phone,…

Appointment(Appointment\_ref, Date, Location,…

With foreign key

Task(Task\_id, Task\_Name)

Project( Project\_id, Project\_StartDate, Project\_EndDate)

Project\_Task(PT\_id, Task\_id, Project\_id, Desc )

Consultant(Consultant\_id, Name, Address, Phone,Project\_id)

Client (Client\_id, Name, Address, Phone,…

Appointment(Appointment\_ref, Date, Location, Client\_id,Consultant\_id)

**C. HOUSE LETS – Normalisation**

The Agency undertakes regular inspections of the properties. Staff are allocated a company pool car for the day. A member of staff may inspect many properties in one day, but a property is inspected only once in any one day.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Page No: 1 | | Acme Letting Agency  Property Inspection Report | | | | | Date: | 12/10/96 | |
| Property No: p103 | | | | Property Address: | | | | | |
| Inspection Date | Inspection Time | | Comments | | Staff Number | Staff Name | | | Car Reg. |
|  |  | |  | |  |  | | |  |
|  |  | |  | |  |  | | |  |

Your task is to identify which Normal Form has been incorrectly normalised.

|  |  |  |  |
| --- | --- | --- | --- |
| **UNF** | **1NF** | **2NF** | **3NF** |
| Property No. | Property No. | Property No. | Property No. |
| Property Addr. | Property Addr. | Property Addr. | Property Addr. |
| Idate\* |  |  |  |
| Itime \* | *Property No.* | *Property No.* | *Property No.* |
| Comment\* | Idate | Idate | Idate |
| Staff\_no\* | Itime | Itime | Itime |
| Sname\* | Comment | Comment | Comment |
| Car\_reg\* | Staff\_no | *Staff\_no* | *Staff\_no* |
|  | Sname |  |  |
|  | Car\_reg |  |  |
|  |  | Staff\_no | Staff\_no |
|  |  | Sname | Sname |
|  |  | Car\_reg | Car\_reg |

The correct form of normalization is

|  |  |  |  |
| --- | --- | --- | --- |
| **UNF** | **1NF** | **2NF** | **3NF** |
| Property No. | Property No. | Property No. | Property No. |
| Property Addr. | Property Addr. | Property Addr. | Property Addr. |
| Idate\* |  |  |  |
| Itime \* | *Property No.* | *Property No.* | *Property No.* |
| Comment\* | Idate | Idate | Idate |
| Staff\_no\* | Itime | Itime | Itime |
| Sname\* | Comment | Comment | Comment |
| Car\_reg\* | Staff\_no | *Staff\_no* | *Staff\_no* |
|  | Sname | Sname |  |
|  | Car\_reg | Car\_reg | *Staff\_no* |
|  |  |  | *name* |
|  |  |  |  |
|  |  |  | Staff\_no |
|  |  |  | Idate |
|  |  |  | Car\_reg |

**Task 3** **- Physical Design**

**Complete the Physical Desing SQL tutorial – based on scott tables.**



Scott Script

DROP TABLE DEPT CASCADE CONSTRAINTS ;

DROP TABLE EMP CASCADE CONSTRAINTS ;

DROP TABLE SALGRADE CASCADE CONSTRAINTS ;

--commands to create tables

CREATE TABLE DEPT (

DEPTNO NUMBER (3) ,

DNAME VARCHAR2 (18) UNIQUE,

LOC VARCHAR2 (17),

CONSTRAINT PK\_DEPT PRIMARY KEY ( DEPTNO ));

CREATE TABLE EMP (

EMPNO NUMBER (5) ,

ENAME VARCHAR2 (14),

JOB VARCHAR2 (10),

MGR NUMBER (5),

HIREDATE DATE,

SAL NUMBER (8,4),

COMM NUMBER (9,3),

DEPTNO NUMBER (6) REFERENCES dept (DEPTNO),

CONSTRAINT PK\_EMP PRIMARY KEY ( EMPNO ));

CREATE TABLE SALGRADE (

GRADE NUMBER,

LOSAL NUMBER,

HISAL NUMBER);

A screenshot of a computer

Description automatically generated

-- insert data into dept, emp and salgrade, note salgrade is not linked to emp and dept, however dept is a FK in emp

INSERT INTO DEPT (DEPTNO, DNAME, LOC) VALUES (5, 'SALES', 'NAIKAP');

INSERT INTO DEPT (DEPTNO, DNAME, LOC) VALUES (10, 'HR', 'KALANKI');

INSERT INTO DEPT (DEPTNO, DNAME, LOC) VALUES (15,'WARRENTY', 'BALKHU');

INSERT INTO DEPT (DEPTNO, DNAME, LOC) VALUES (20,'ADMIN', 'SITAPAILA');

-- Employee 1

INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) VALUES (1001, 'JOHN', 'ENGINEER', 5678, '01/10/2023', 800, 200, 40);

-- Employee 2

INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) VALUES (7501, 'EMMA', 'CLERK', 7782, '03/15/2013', 1200, 150, 20);

-- Employee 3

INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) VALUES (7522, 'LUCY', 'ANALYST', 7788, '07/07/2014', 1800, 400, 10);

-- Employee 4

INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) VALUES (7567, 'DAVID', 'MANAGER', 7839, '09/30/2012', 3500, null, 30);

-- Employee 5

INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) VALUES (7655, 'SARAH', 'SALESMAN', 7698, '04/12/2015', 1450, 200, 30);

-- Employee 6

INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) VALUES (7700, 'RYAN', 'CLERK', 7782, '11/20/2013', 2200, null, 10);

-- Employee 7

INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) VALUES (7783, 'MIA', 'CLERK', 7782, '02/18/2014', 1700, null, 10);

-- Employee 8

INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) VALUES (7790, 'OLIVER', 'ANALYST', 7566, '08/25/2018', 3200, null, 20);

-- Employee 9

INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) VALUES (7840, 'AVA', 'MANAGER', 7839, '01/05/2014', 4800, null, 10);

-- Employee 10

INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) VALUES (7845, 'NOAH', 'SALESMAN', 7698, '06/30/2016', 1700, 100, 30);

INSERT INTO salgrade(GRADE, LOSAL,HISAL) VALUES (01, 2300, 1000);

INSERT INTO salgrade(GRADE, LOSAL,HISAL) VALUES (02, 3401, 2000);

INSERT INTO salgrade(GRADE, LOSAL,HISAL) VALUES (03, 1344, 3000);

INSERT INTO salgrade(GRADE, LOSAL,HISAL) VALUES (04, 3456, 4000);

INSERT INTO salgrade(GRADE, LOSAL,HISAL) VALUES (05, 4589, 5000);

A screenshot of a computer

Description automatically generated

-- this command will add FK constraint to emp table, based on the recursive relationship

ALTER TABLE emp ADD CONSTRAINT fk\_manager FOREIGN KEY (MGR)REFERENCES emp(EMPNO);

-- this command is to create and drop sequences for PK columns in emp and dept tables

DROP sequence dept\_seq;

CREATE sequence dept\_seq start with 50;

DROP sequence emp\_seq;

CREATE sequence emp\_seq start with 8000;

COMMIT;

create or replace trigger emp\_T1

BEFORE insert on emp

for each row

begin

IF :NEW.empno IS NULL THEN

SELECT emp\_SEQ.NEXTVAL INTO :NEW.empno FROM SYS.DUAL;

END IF;

end;

/

commit;

create or replace trigger dept\_T1

BEFORE

insert on dept

for each row

begin

if :NEW.deptno is null then

select dept\_SEQ.nextval into :NEW.deptno from sys.dual;

end if;

end;

/

commit;

/

A screenshot of a computer

Description automatically generated

1. Notice departments are in different cities. (horizontal partitioning)
   1. Write SQL to create 4 ‘emp’ tables, one for each city eg tmpempny, tmpempda. Check you haven’t lost any data. Create this as a script (drop tables before creating to make it re-runnable).

-- Drop tables if they exist

DROP TABLE IF EXISTS tmpempny;

DROP TABLE IF EXISTS tmpempda;

DROP TABLE IF EXISTS tmpempla;

DROP TABLE IF EXISTS tmpempsf;

-- Create tmpempny table

CREATE TABLE tmpempny (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(255),

emp\_salary DECIMAL(10, 2),

emp\_city VARCHAR(50)

);

-- Create tmpempda table

CREATE TABLE tmpempda (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(255),

emp\_salary DECIMAL(10, 2),

emp\_city VARCHAR(50)

);

-- Create tmpempla table

CREATE TABLE tmpempla (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(225),

emp\_salary DECIMAL(10, 2),

emp\_city VARCHAR(50)

);

-- Create tmpempsf table

CREATE TABLE tmpempsf (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(255),

emp\_salary DECIMAL(10, 2),

emp\_city VARCHAR(50)

);

-- Check if tables are created successfully

SELECT \* FROM tmpempny;

SELECT \* FROM tmpempda;

SELECT \* FROM tmpempla;

SELECT \* FROM tmpempsf;

-- Insert data into tmpempny

INSERT INTO tmpempny (emp\_id, emp\_name, emp\_salary, emp\_city)

VALUES (1, 'John Kangaroo', 50000.00, 'New York');

INSERT INTO tmpempny (emp\_id, emp\_name, emp\_salary, emp\_city)

VALUES (2, 'Jane Smith', 60000.00, 'New York');

INSERT INTO tmpempny (emp\_id, emp\_name, emp\_salary, emp\_city)

VALUES (3, 'Bob Johnson', 55000.00, 'New York');

-- Insert data into tmpempda

INSERT INTO tmpempda (emp\_id, emp\_name, emp\_salary, emp\_city)

VALUES (4, 'Alice Brown', 52000.00, 'Dallas');

INSERT INTO tmpempda (emp\_id, emp\_name, emp\_salary, emp\_city)

VALUES(5, 'Charlie Davis', 48000.00, 'Dallas');

INSERT INTO tmpempda (emp\_id, emp\_name, emp\_salary, emp\_city)

VALUES(6, 'David White', 51000.00, 'Dallas');

-- Insert data into tmpempla

INSERT INTO tmpempla (emp\_id, emp\_name, emp\_salary, emp\_city)

VALUES(7, 'Eva Martinez', 53000.00, 'Los Angeles');

INSERT INTO tmpempla (emp\_id, emp\_name, emp\_salary, emp\_city)

VALUES(8, 'Frank Garcia', 59000.00, 'Los Angeles');

INSERT INTO tmpempla (emp\_id, emp\_name, emp\_salary, emp\_city)

VALUES (9, 'Grace Rodriguez', 56000.00, 'Los Angeles');

-- Insert data into tmpempsf

INSERT INTO tmpempsf (emp\_id, emp\_name, emp\_salary, emp\_city)

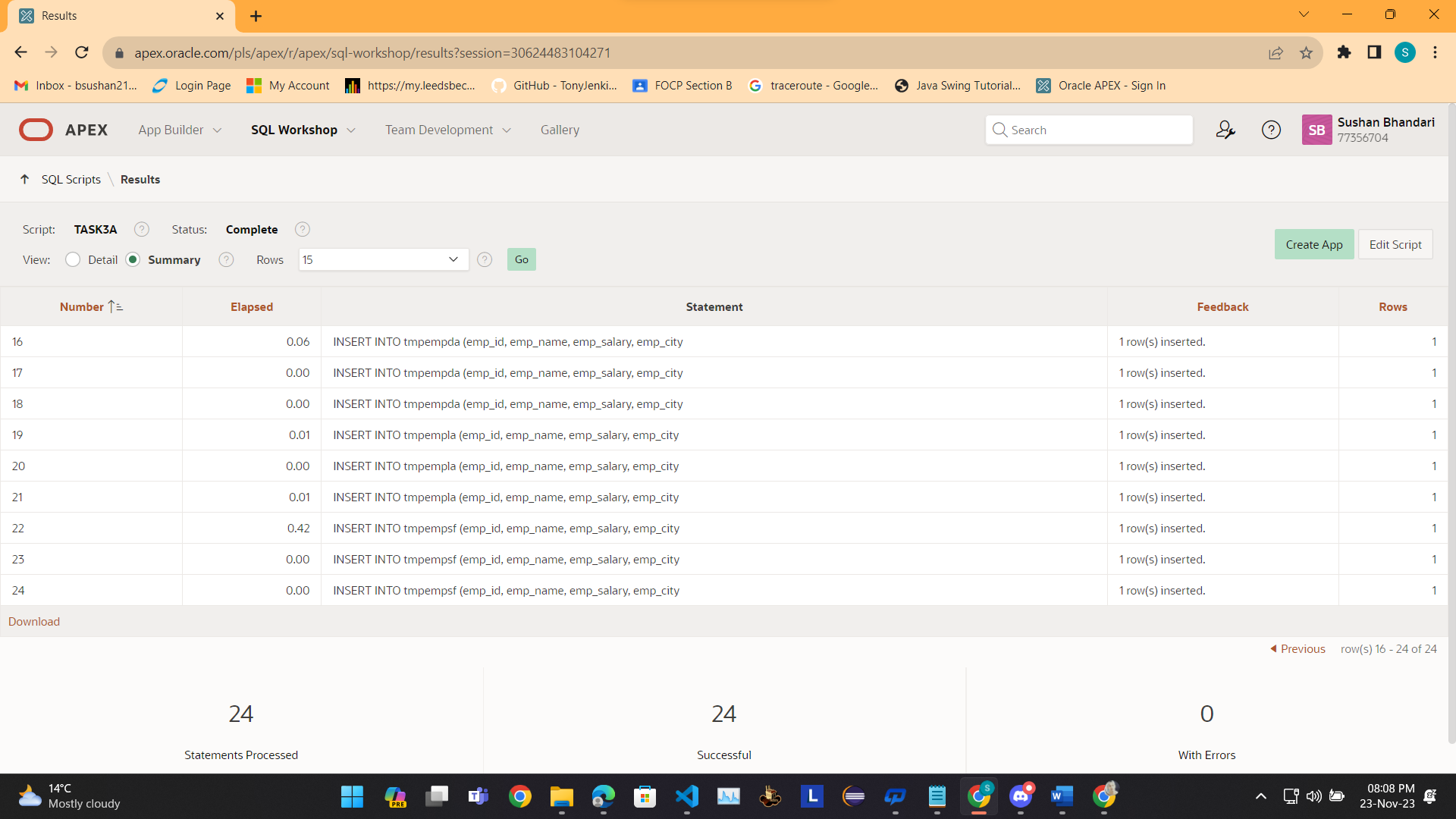
VALUES (10, 'Henry Lee', 54000.00, 'San Francisco');

INSERT INTO tmpempsf (emp\_id, emp\_name, emp\_salary, emp\_city)

VALUES (11, 'Isabel Kim', 58000.00, 'San Francisco');

INSERT INTO tmpempsf (emp\_id, emp\_name, emp\_salary, emp\_city)

VALUES (12, 'Jack Wilson', 57000.00, 'San Francisco');



* 1. Now write SQL to select all employees from the Company.

Hint: use the UNION command. UNION compatibility is important.

-- Select all employees from the Company using UNION

SELECT emp\_id, emp\_name, emp\_salary, emp\_city FROM tmpempny

UNION

SELECT emp\_id, emp\_name, emp\_salary, emp\_city FROM tmpempda

UNION

SELECT emp\_id, emp\_name, emp\_salary, emp\_city FROM tmpempla

UNION

SELECT emp\_id, emp\_name, emp\_salary, emp\_city FROM tmpempsf;

A screenshot of a computer

Description automatically generated

1. Assume we need to hold sensitive employee data on one table ‘tmpempprsnnl’ and non sensitive data on another table tmpempcommon.

Write the SQL to do this. Check the data is correct. Create it as a re-runnable script. (vertical partitioning).

-- Drop tables if they exist

DROP TABLE tmpempprsnnl;

DROP TABLE tmpempcommon;

-- Create tmpempprsnnl table for sensitive data

CREATE TABLE tmpempprsnnl (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(255),

emp\_salary DECIMAL(10, 2),

emp\_city VARCHAR(50)

);

-- Create tmpempcommon table for non-sensitive data

CREATE TABLE tmpempcommon (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(255),

emp\_city VARCHAR(50)

);

-- Insert sample data into tmpempprsnnl

INSERT INTO tmpempprsnnl (emp\_id, emp\_name, emp\_salary, emp\_city)

VALUES(1, 'John Doe', 50000.00, 'New York');

INSERT INTO tmpempprsnnl (emp\_id, emp\_name, emp\_salary, emp\_city)

VALUES(2, 'Jane Smith', 60000.00, 'New York');

-- Insert sample data into tmpempcommon

INSERT INTO tmpempcommon (emp\_id, emp\_name, emp\_city)

VALUES(1, 'John Doe', 'New York');

INSERT INTO tmpempcommon (emp\_id, emp\_name, emp\_city)

VALUES(2, 'Jane Smith', 'New York');

-- Check data in tmpempprsnnl

SELECT \* FROM tmpempprsnnl;

-- Check data in tmpempcommon

SELECT \* FROM tmpempcommon;

A screenshot of a computer

Description automatically generated

1. 2a) Now write SQL to select all details of all employees from tmpempprsnnl and tmpempcommon.

-- Select all details of all employees from tmpempprsnnl and tmpempcommon

SELECT emp\_id, emp\_name, emp\_salary, emp\_city FROM tmpempprsnnl

UNION

SELECT emp\_id, emp\_name, NULL AS emp\_salary, emp\_city FROM tmpempcommon;

A screenshot of a computer

Description automatically generated

1. Do a combination of the above. There should be one tmpempprsnnl table containing all the sensitive data for all the employees and 4 other emp tables containing the general emp data (tmpempny, tmpempda etc). This is combined partitioning.

-- Drop tables if they exist

DROP TABLE IF EXISTS tmpempprsnnl, tmpempny, tmpempda, tmpempla, tmpempchi;

-- Create tmpempprsnnl for sensitive data

CREATE TABLE tmpempprsnnl AS

SELECT emp\_id, emp\_name, emp\_ssn, emp\_dob FROM emp;

-- Create tmpempny for New York

CREATE TABLE tmpempny AS

SELECT emp\_id, emp\_salary, emp\_position FROM emp WHERE department = 'New York';

-- Create tmpempda for Dallas

CREATE TABLE tmpempda AS

SELECT emp\_id, emp\_salary, emp\_position FROM emp WHERE department = 'Dallas';

-- Create tmpempla for Los Angeles

CREATE TABLE tmpempla AS

SELECT emp\_id, emp\_salary, emp\_position FROM emp WHERE department = 'Los Angeles';

-- Create tmpempchi for Chicago

CREATE TABLE tmpempchi AS

SELECT emp\_id, emp\_salary, emp\_position FROM emp WHERE department = 'Chicago';