**The British College**

**KATHMANDU**

**Coursework Submission Coversheet**(individual coursework only)

**Faculty of Arts, Environment and Technology LBU Student Id:**

77261075

**For checking by the student:**

Please ensure all information is complete and correct and attach this form securely to the front of your work before posting it in a coursework collection box.

Award name: Bsc Hons in Computing

Module code: [15781 - AUT - 202210](https://my.leedsbeckett.ac.uk/webapps/blackboard/execute/courseMain?course_id=_157454_1)

Module name: Advance Database System (ADS)

Module run: 2022

Coursework title: Data warehouse Design and Development

Due Date:

Module leader: (In LBU): Jackie Campbell, Sanela Lazarevski

Module tutor: (In TBC): Dibya Tara Shakya

**TURNITIN** Checked: YES NO ***(please circle)***

Submission date& time: Date: Time: Before noon

**Total Word Count: Total Number of Pages (including this front sheet):**

**In submitting this form with your assignment, you make the following declaration:**  
I declare, that the coursework submitted is my own work and has not (either in whole or part) been submitted towards the award of any other qualification either at LBU or elsewhere. I have fully attributed/referenced all sources of information used during the completion of my assignment, and I am aware that failure to do so constitutes an assessment offence.

Signed: Mavira Bhattarai Date: <Submitted Date>

**You are strongly advised to retain a second copy of your work in case of any query about the assignment.**

**For completion by the faculty:**

**This mark is provisional and subject to moderation and approval by the relevant examining board**

**Teacher's Feedback**

**Teacher's Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Contents

[Task 2: Take your design and code further by using PL/SQL techniques 3](#_Toc125362916)

[Identify, code, test a function, procedure and package to support the ETL task. 3](#_Toc125362917)

[1. Overview of PL/SQL 3](#_Toc125362918)

[2. Aspects of PL/SQL: 3](#_Toc125362919)

[1. Creating Packages for PL/SQL 3](#_Toc125362920)

[For Staging Tables 4](#_Toc125362921)

[For Bad Stage Tables 6](#_Toc125362922)

[For Recleaning Data 10](#_Toc125362923)

[For transform table 13](#_Toc125362924)

[FOR DIM\_TIME 15](#_Toc125362925)

[2. Inserting from data source PRCS 17](#_Toc125362926)

[a) Insert into stage table 17](#_Toc125362927)

# Task 2: Take your design and code further by using PL/SQL techniques

## Identify, code, test a function, procedure and package to support the ETL task.

### Overview of PL/SQL

Programmers can combine the features of SQL with procedural statements by using PL/SQL, a block structured language. All of a block's assertions are submitted at once to the Oracle engine, which speeds up processing and lowers bandwidth. By using SQL and PL/procedural SQL's approach, you may instruct the compiler on both "what to do" and "how to do it." Similar to other database languages, it makes use of loops, conditions, and object-oriented concepts to provide programmers more control. The full name of PL/SQL is "Procedural Language extensions to SQL".

### Aspects of PL/SQL:

1. PL/SQL, in general, is a procedural language that provides decision-making, iteration, and many other features typical of procedural programming languages.
2. Multiple queries can be run in a block using a single PL/SQL query.
3. PL/SQL units like as procedures, functions, packages, triggers, and types can be developed and saved in the database for later usage by applications.
4. The PL/SQL feature known as the exception handling block enables you to handle exceptions that occur inside PL/SQL blocks.
5. PL/SQL applications can run on any piece of hardware or operating system that supports Oracle.
6. PL/SQL has comprehensive error checking.

# 1. Creating Packages for PL/SQL

All the procedures are kept inside packages

## For Staging Tables

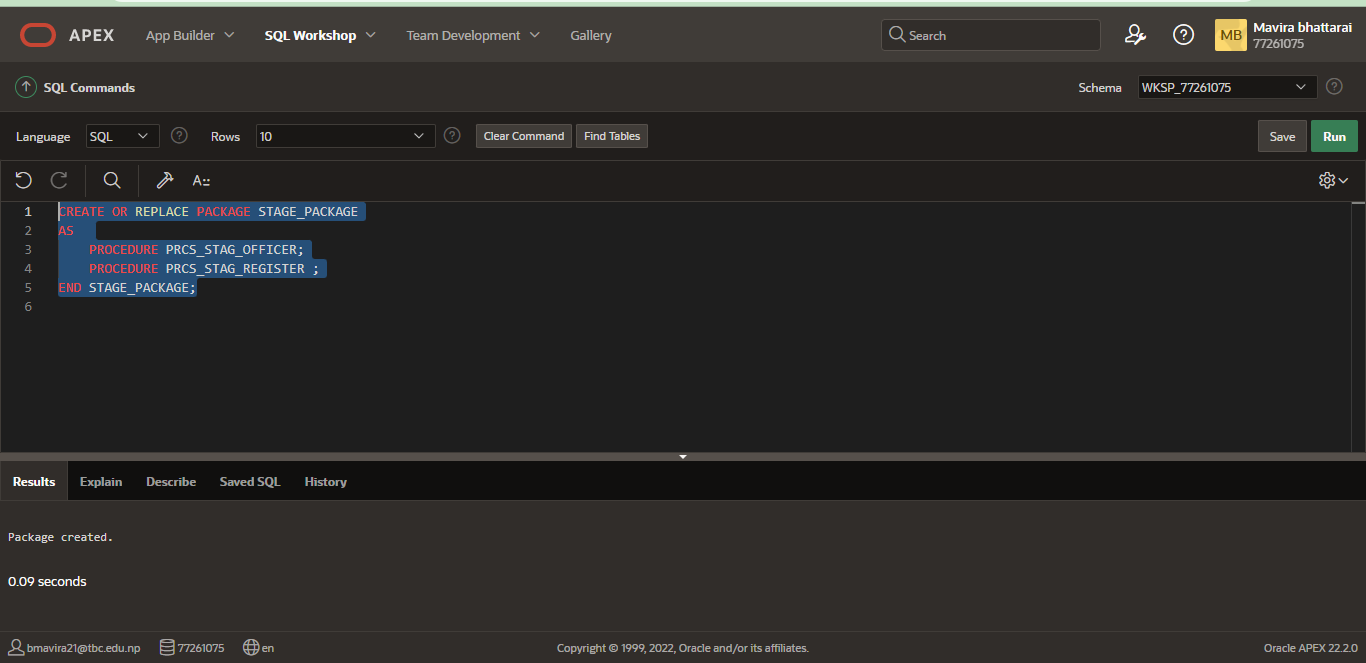


Figure: Creating Package

A Package is created for procedures used for staging Table.

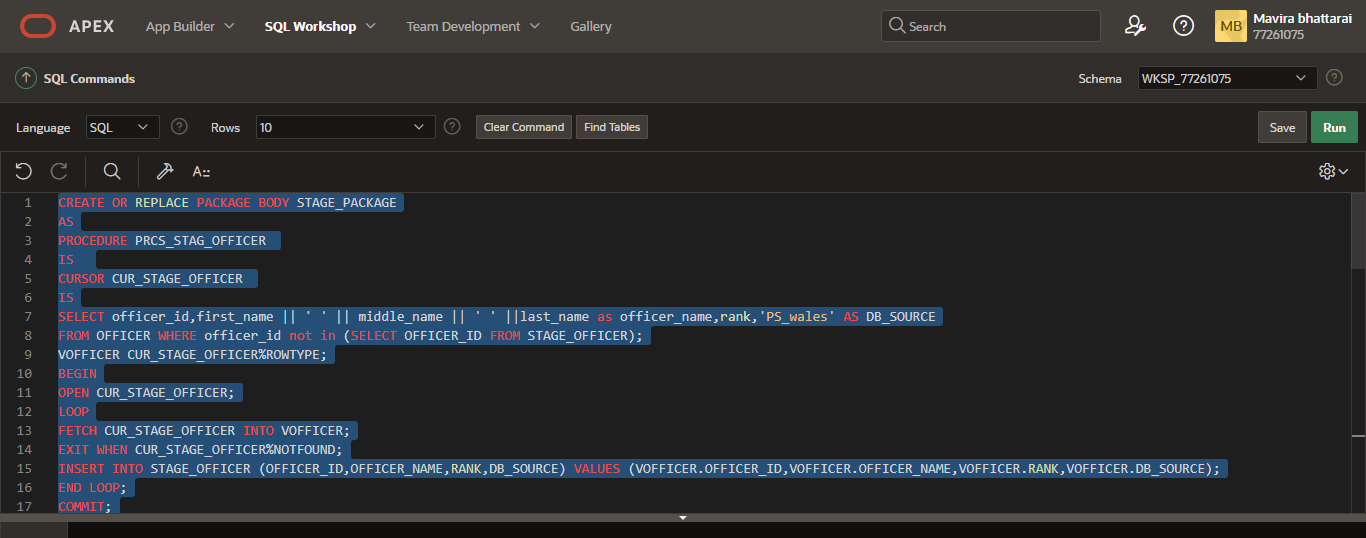


Figure: Making Procedure for STAGE\_OFFICER

A procedure PRCS\_STAG\_OFFICER is created to insert all the data of officer from OFFICER table of PS\_wales Data Source

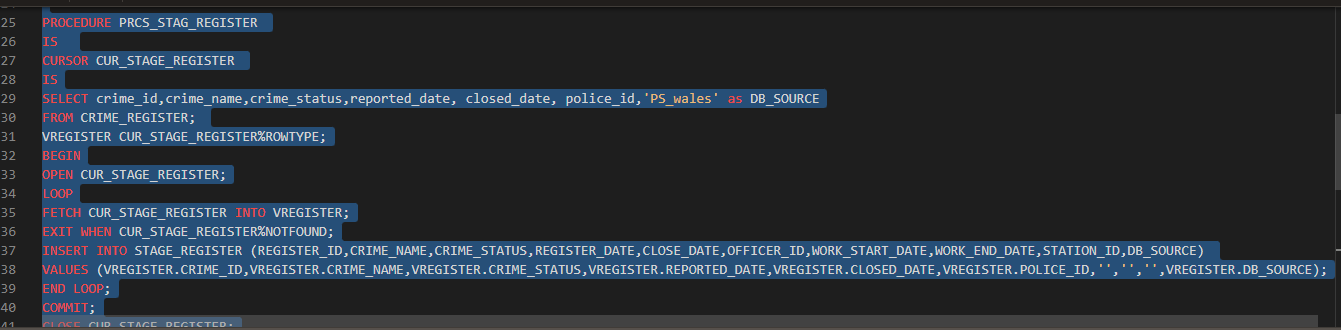


Figure: Making Procedure for STAGE\_REGISTER

A procedure PRCS\_STAG\_REGISTER is created to insert all the data of officer from CRIME\_REGISTER of PS\_wales Data Source

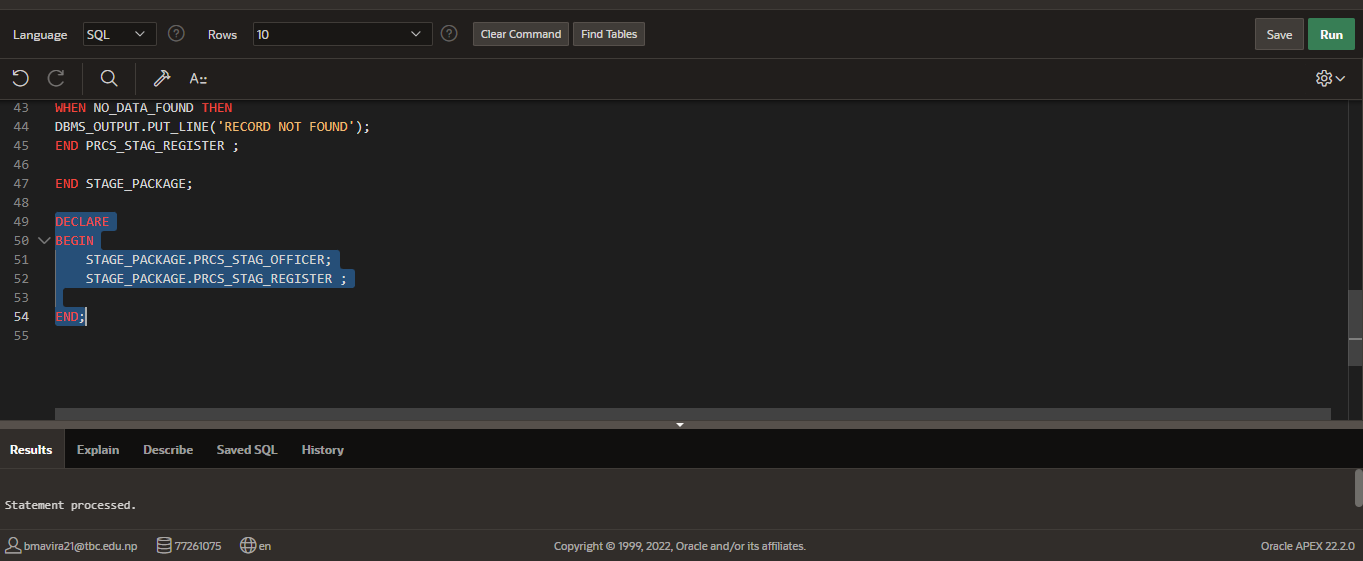


Figure: DECLARING AND RUNNING PACKAGE

The package is declared and run.

## For Bad Stage Tables

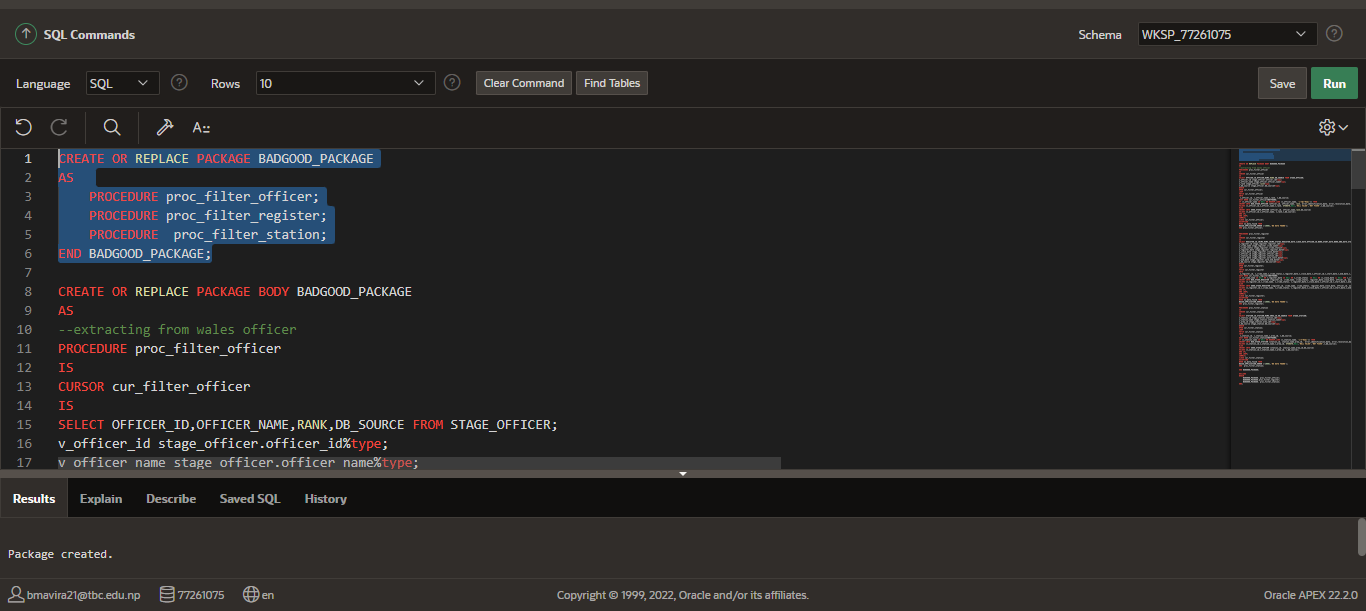
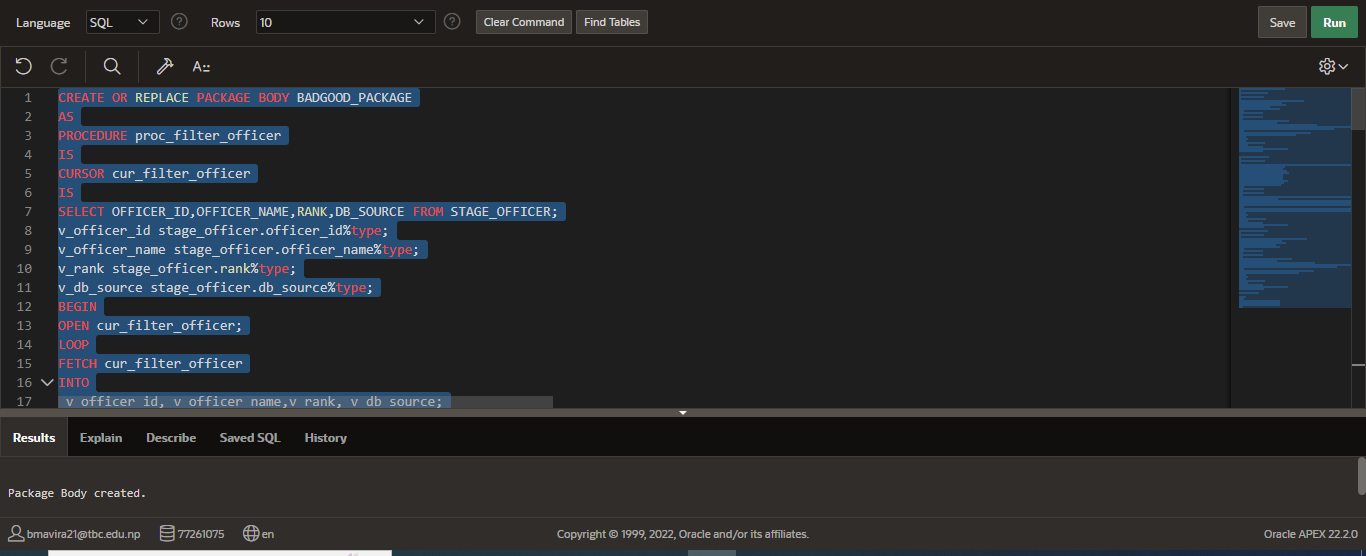
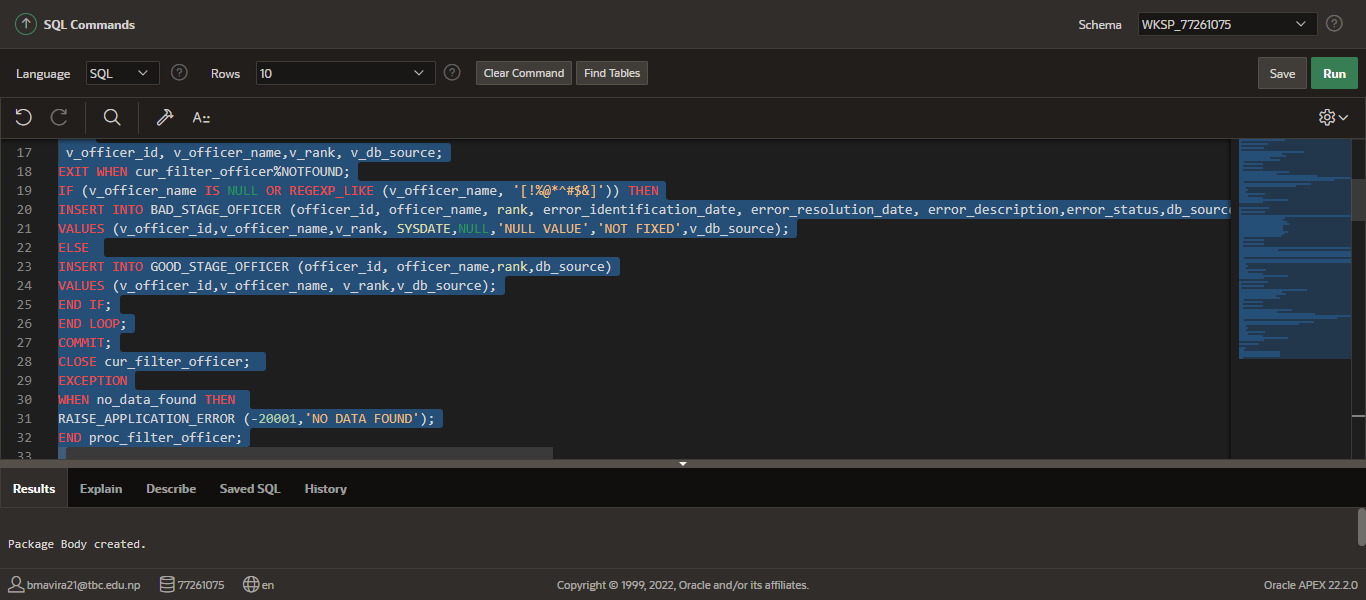


Figure: CREATING PACKAGE FOR BAD GOOD TABLE

A package is created where att the procedures made for inserting data into bad and good tables are kept.

Figure: Making Procedure for GOOD\_STAGE\_OFFICER AND BAD\_STAGE\_OFFICER

A Procedure proc\_filter\_officer is made for inserting all the data of officer from STAGE\_OFFICER. The bulk insert is done using cursor cur\_filter\_officer.



If the data is good it goes to the GOOD\_STAGE\_REGISTER. However, if the data is null or missing or invalid, it goes the BAD\_STAGE\_REGISTER.

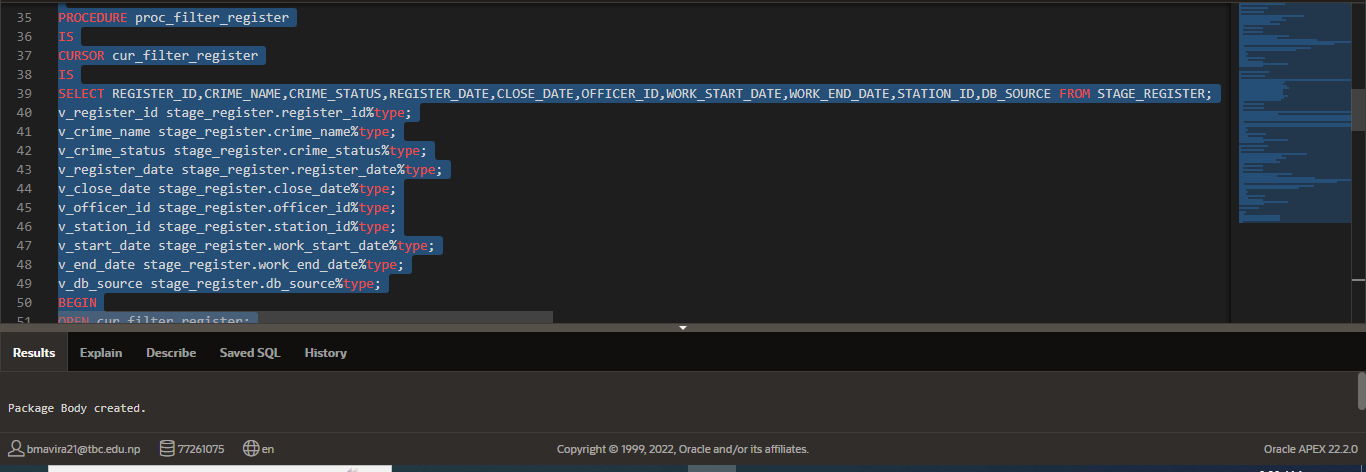
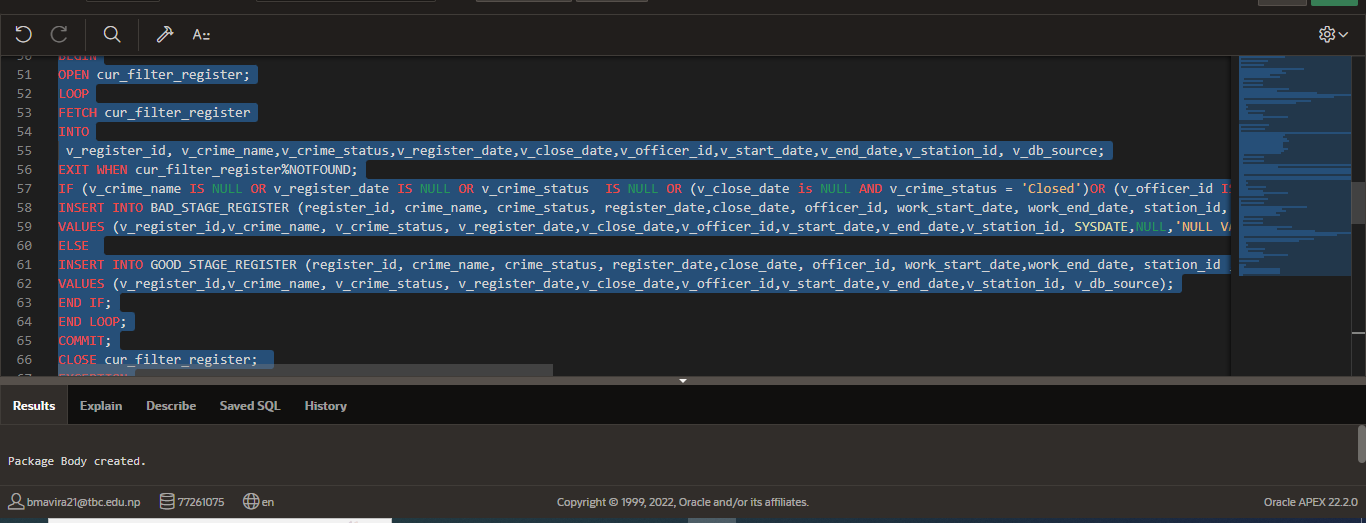


Figure: Making Procedure for GOOD\_STAGE\_REGISTER AND BAD\_STAGE\_REGISTER

A Procedure proc\_filter\_register is made for inserting all the data of officer from STAGE\_REGISTER. The bulk insert is done using cursor cur\_filter\_register.



If the data is good it goes to the GOOD\_STAGE\_REGISTER. However, if the data is null or missing or invalid, it goes the BAD\_STAGE\_REGISTER.

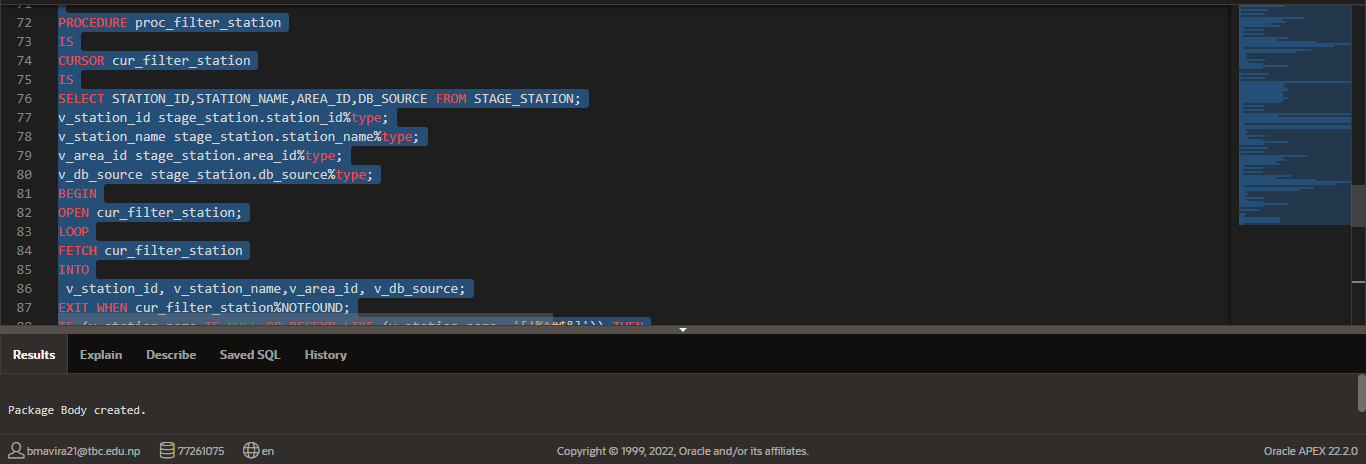
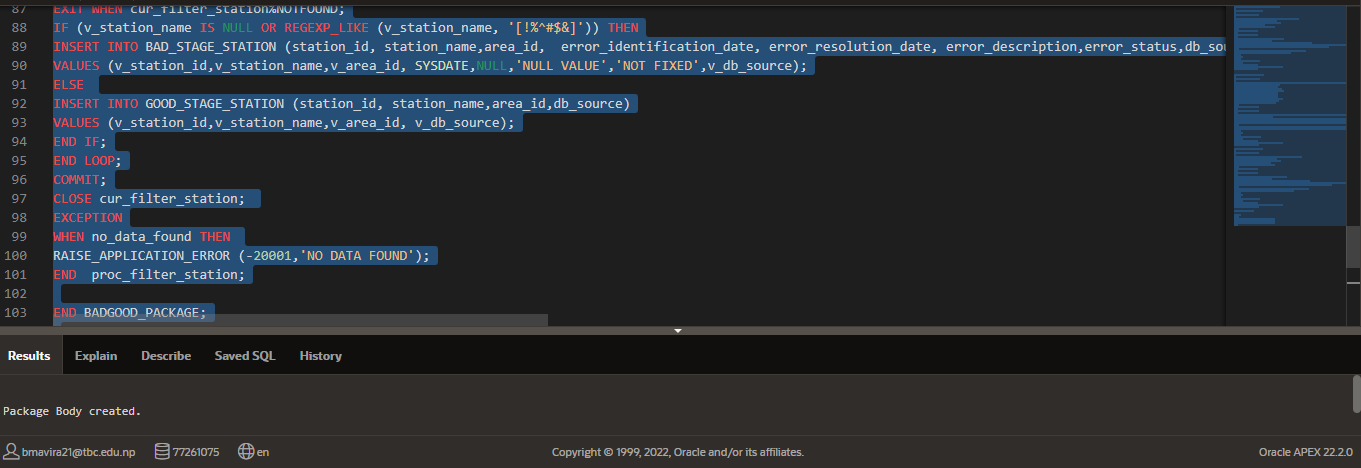


Figure: Making Procedure for GOOD\_STAGE\_STATION AND BAD\_STAGE\_STATION

A Procedure proc\_filter\_station is made for inserting all the data of officer from STAGE\_STATION. The bulk insert is done using cursor cur\_filter\_station.



If the data is good it goes to the GOOD\_STAGE\_STATION. However, if the data is null or missing or invalid, it goes the BAD\_STAGE\_STATION.

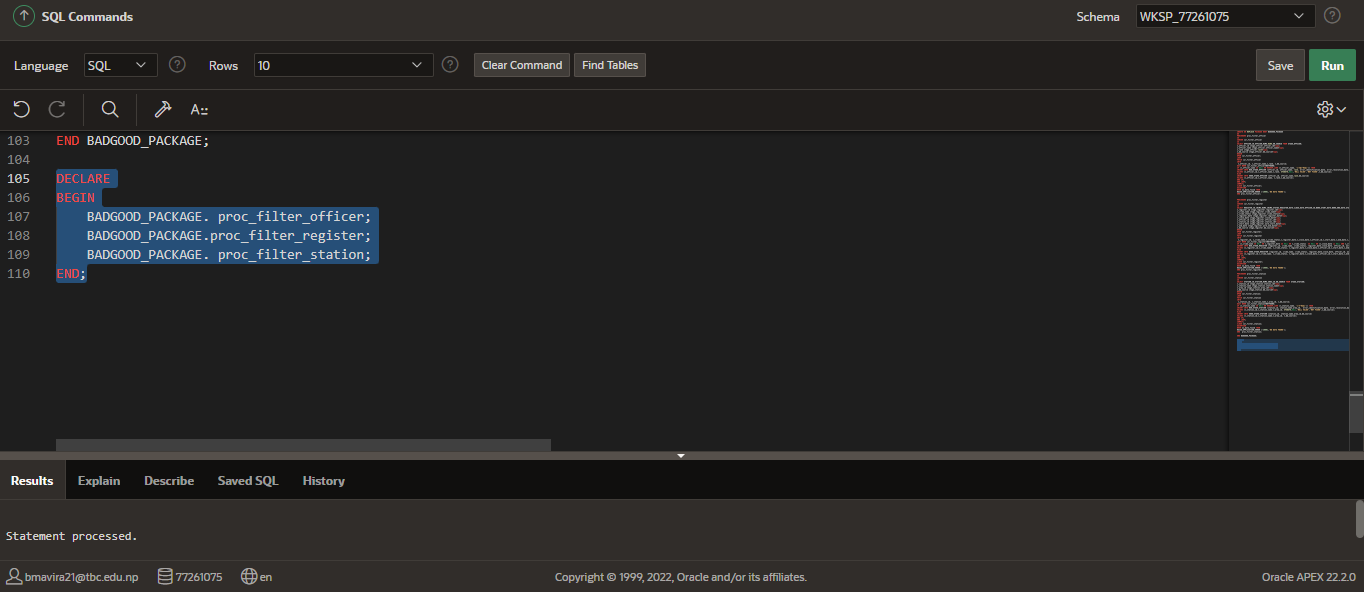


Figure: DECLARING AND RUNNING PACKAGE

Calling the procedure using the package name. It inserts data into the tables after calling the procedures.

## For Recleaning Data

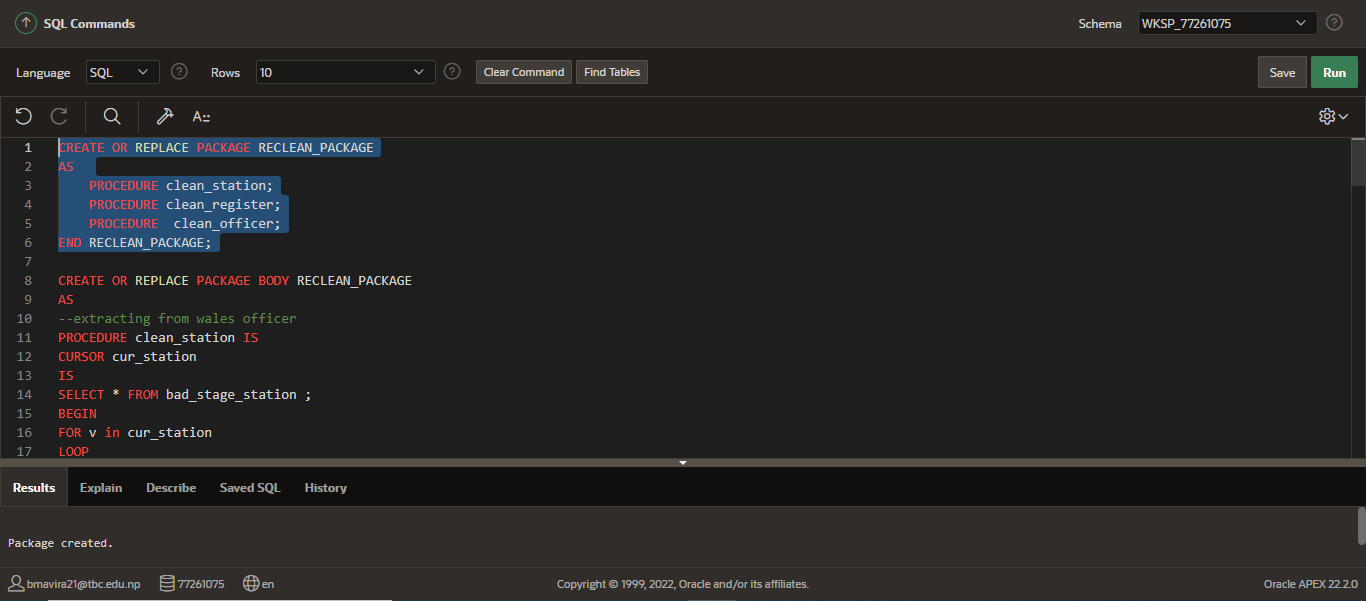


Figure: Creating Package for recleaning process

Package declared to reclean data.

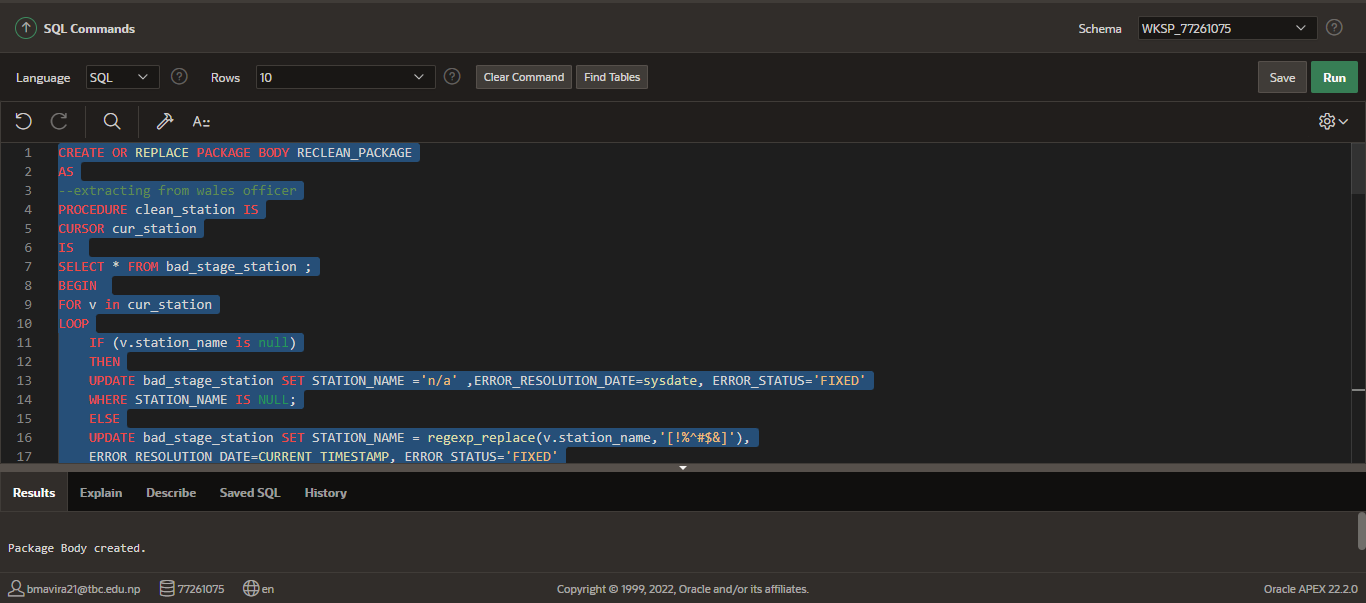


Figure: Making Procedure to clean data from BAD\_STAGE\_STATION

Package body declared for recleaning the bad data of station. This package recleans the bad data and merges into the good data table of station.

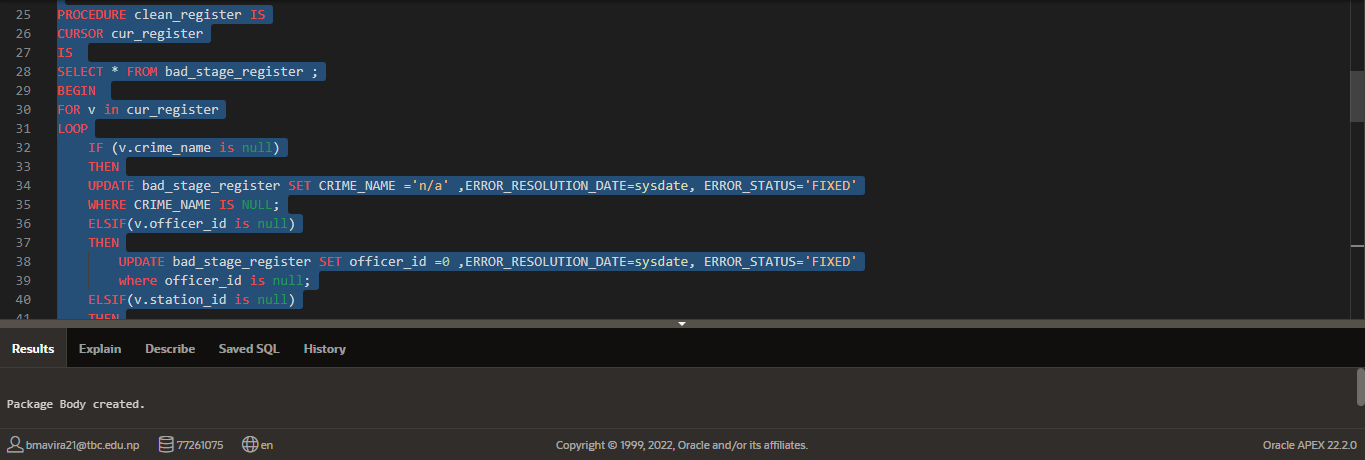
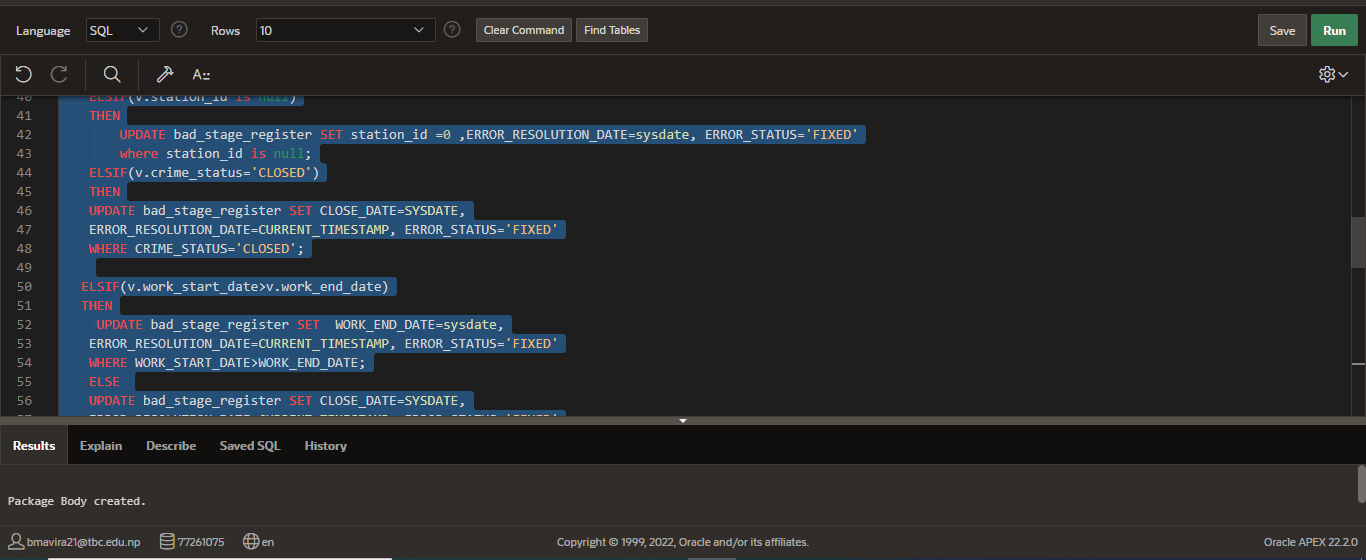
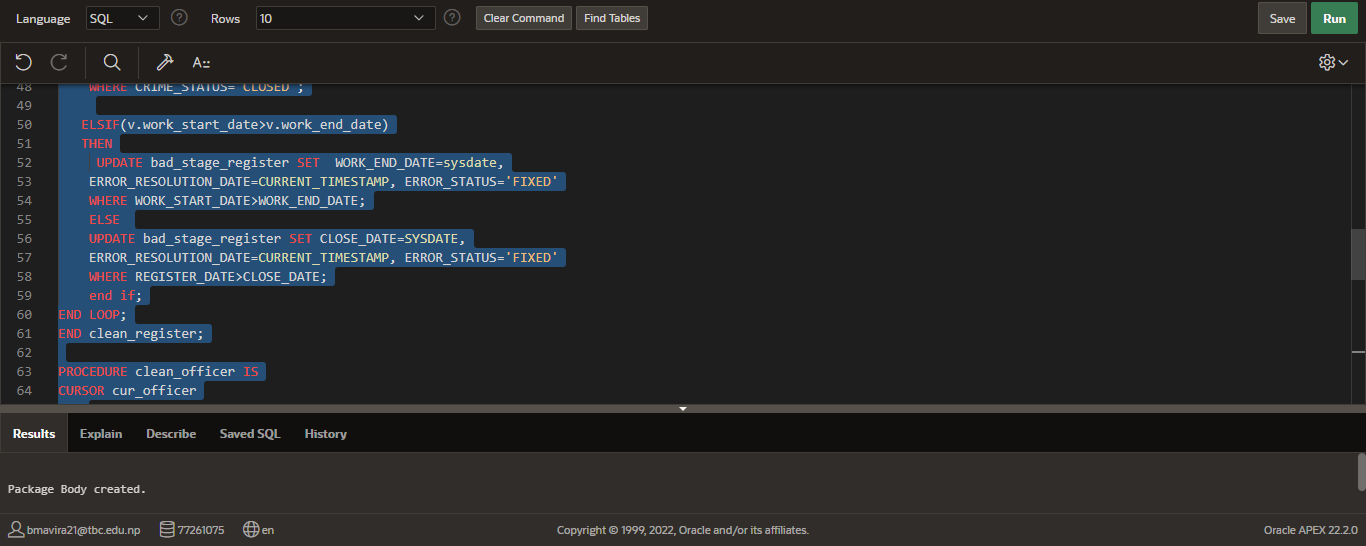


Figure: Making Procedure to clean data from BAD\_STAGE\_REGISTER





Package body declared for recleaning the bad data of crime. This package recleans the bad data and merges into the good data table of crime into good\_stage\_register.

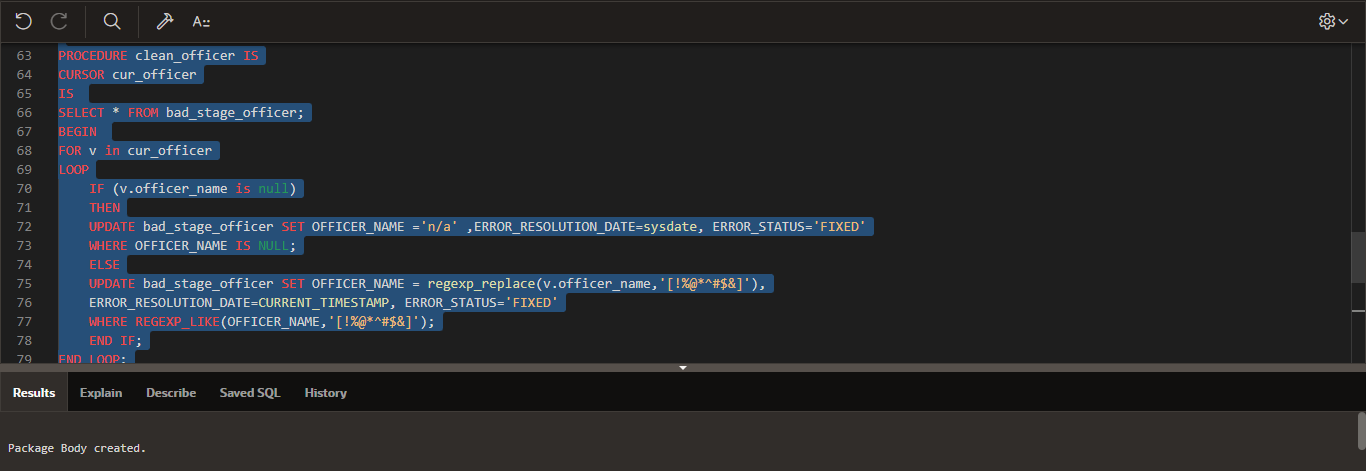


Figure: Making Procedure to clean data from BAD\_STAGE\_OFFICER

Package body declared for recleaning the bad data of officer. This package recleans the bad data and merges into the good data table of officer.

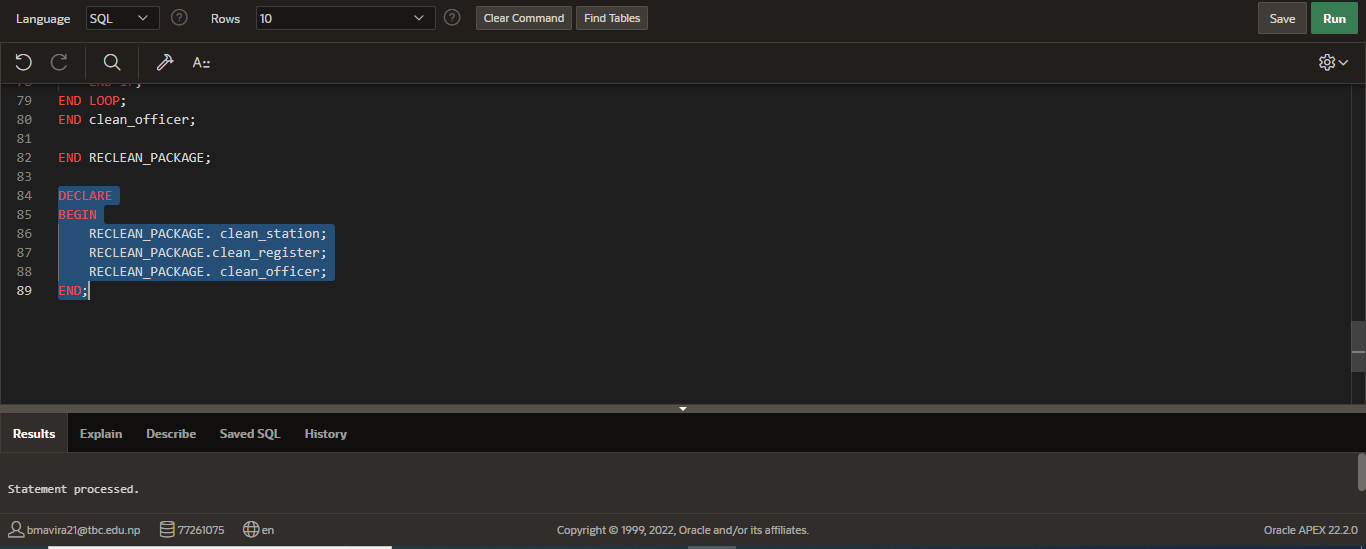


Figure: DECLARING AND RUNNING PACKAGE

Calling the procedure using package name.

## For transform table

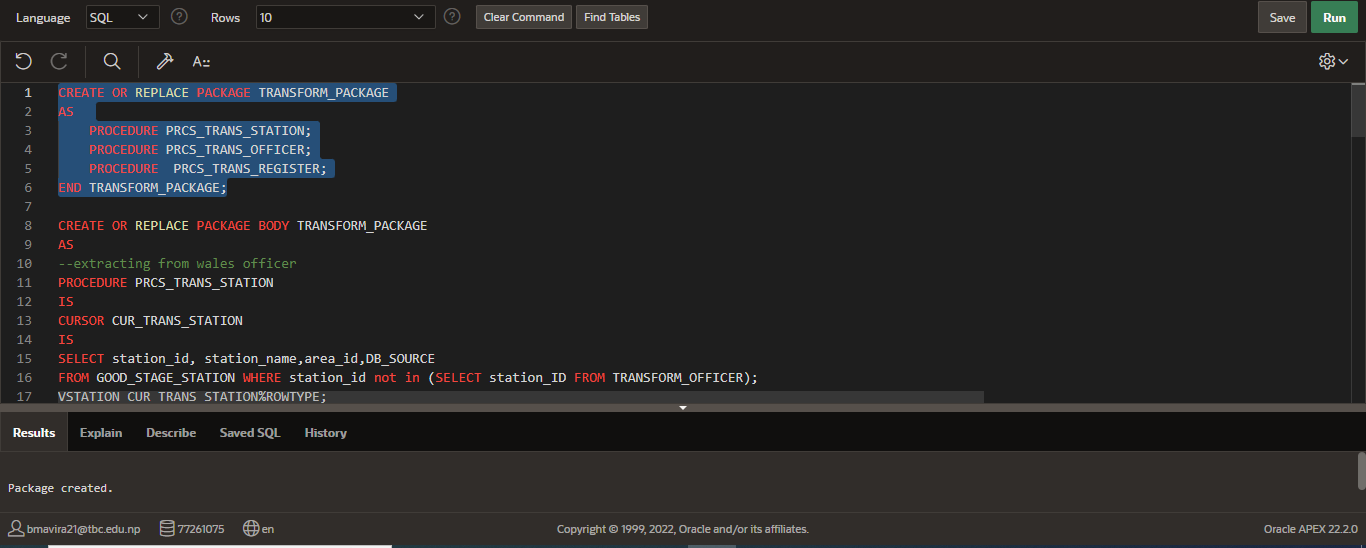


Figure: CREATING A PACKAGE FOR TRANSFORM TABLE

Package name declaration for transform table.

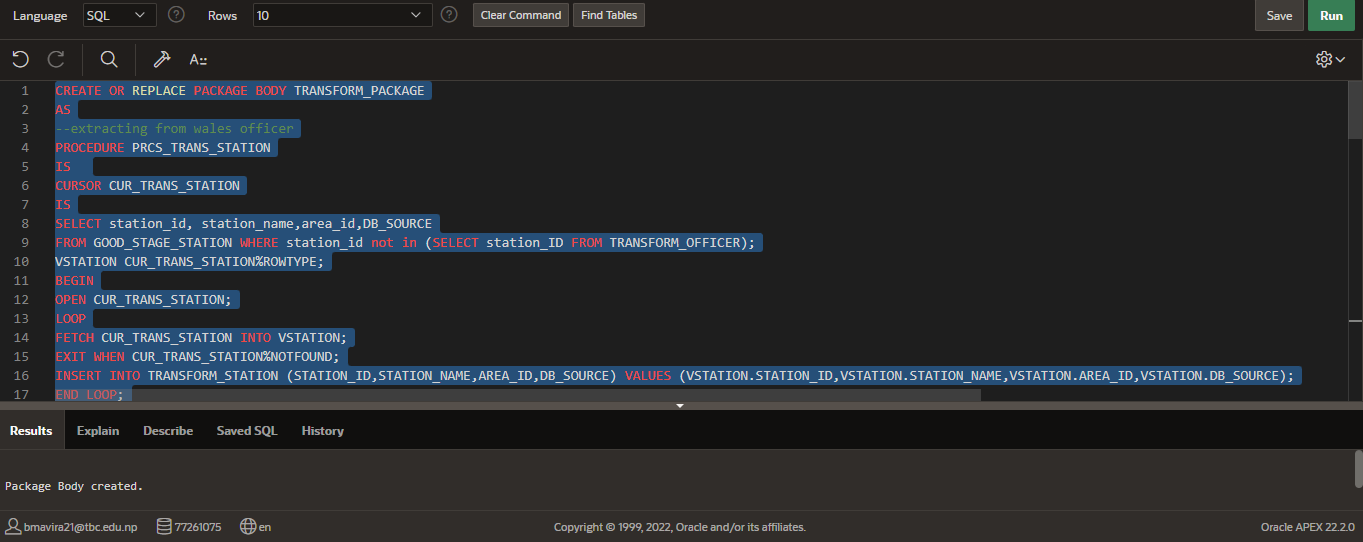


Figure: Making Procedure to transform data from BAD\_STAGE\_STATION AND GOOD\_STAGE\_STATION

Package body declaration for transformation table. Declaring package body for transformation table of station to insert data into the transformation table of station.

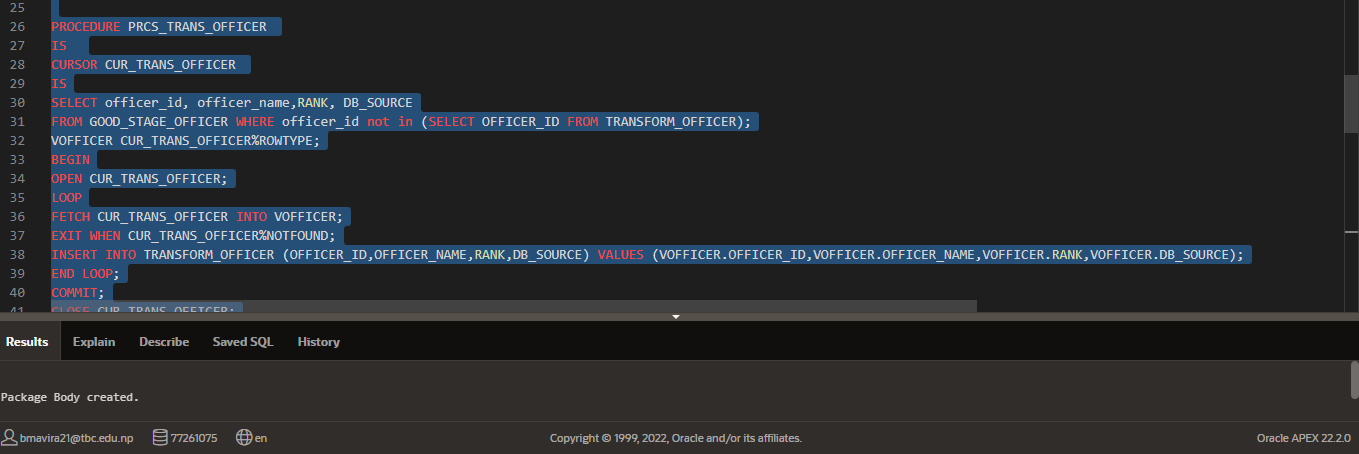


Figure: Making Procedure to transform data from BAD\_STAGE\_OFFICER AND GOOD\_STAGE\_OFFICER

Package body declaration for transformation table. Declaring package body for transformation table of officer to insert data into the transformation table of officer.

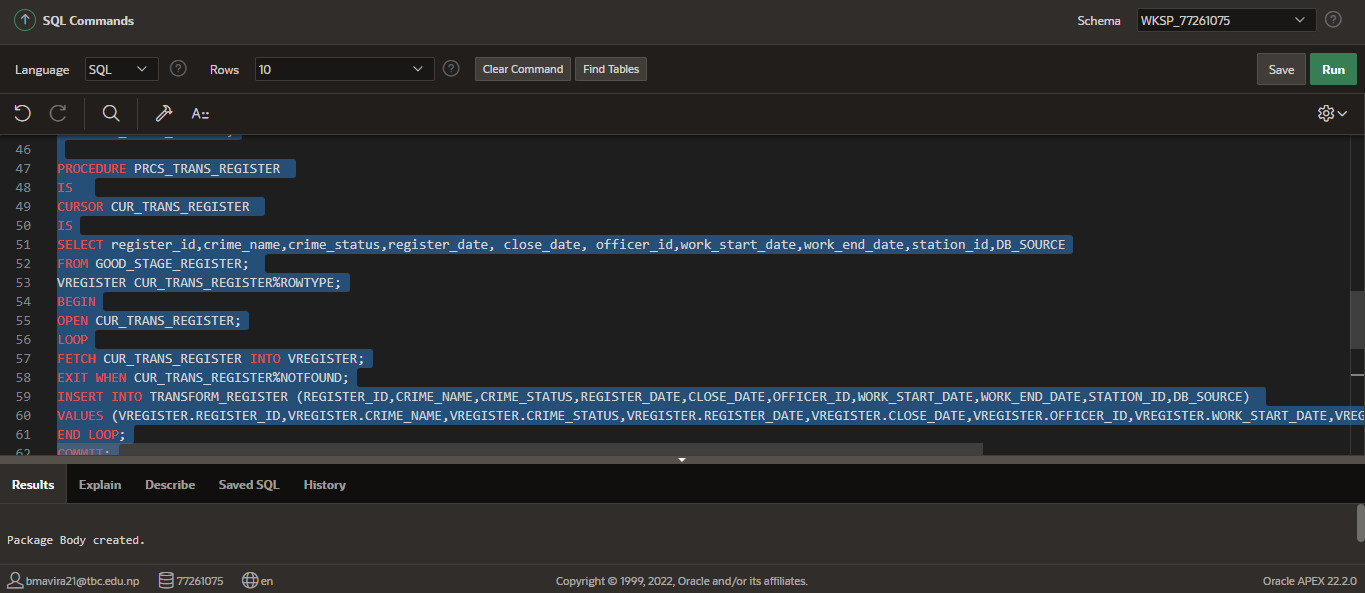


Figure: Making Procedure to transform data from BAD\_STAGE\_REGISTER AND GOOD\_STAGE\_REGISTER

Package body declaration for transformation table. Declaring package body for transformation table of crime to insert data into the transformation table of crime.

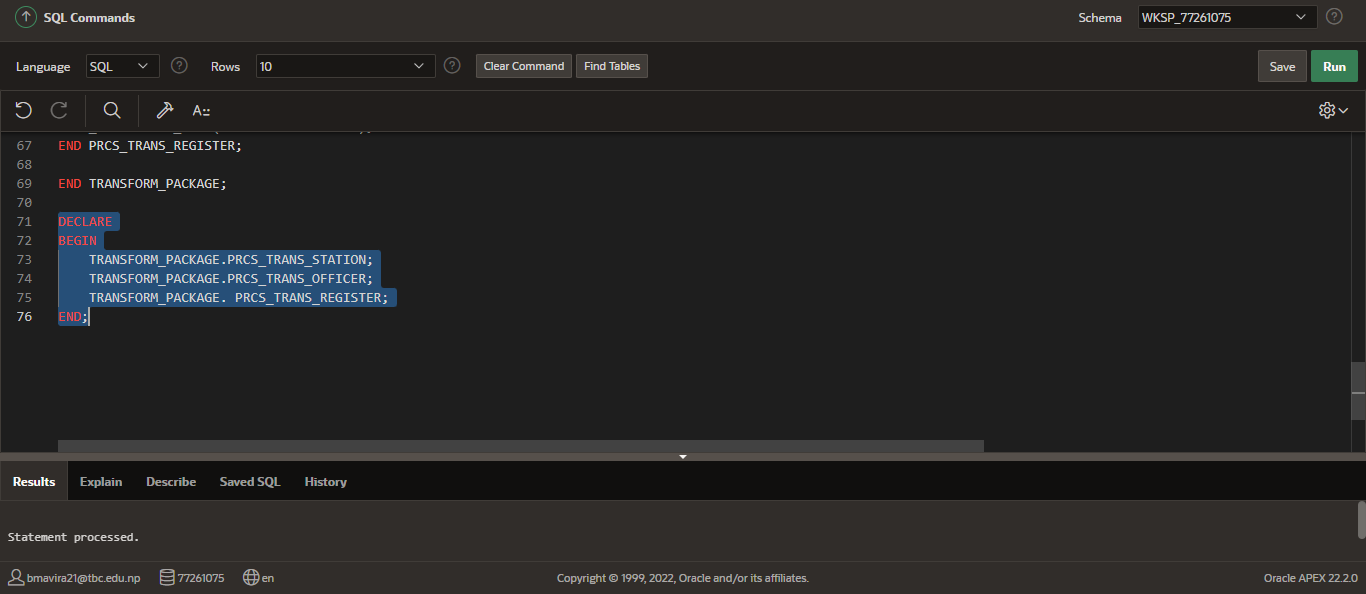


Figure: Declaring and Running package for transform tables

Calling the package for transformation table. This inserts data into the transformation tables.

## FOR DIM\_TIME

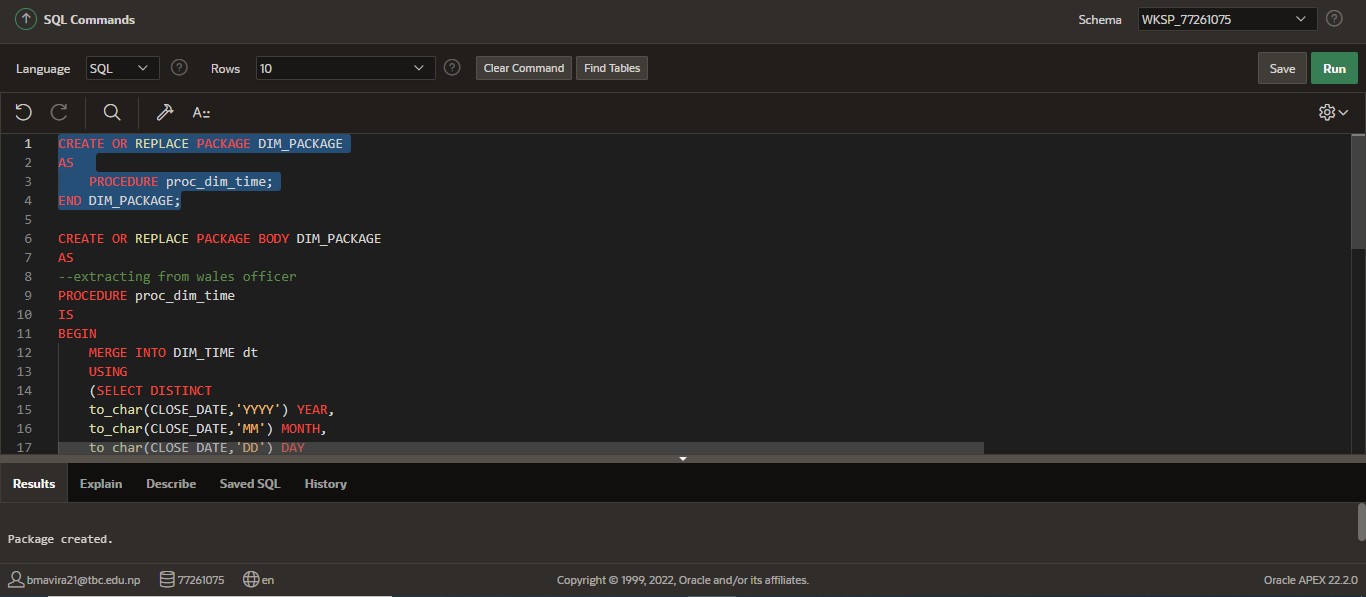


Figure: Creating Package for dim table Time

Package declaration for dim time table.

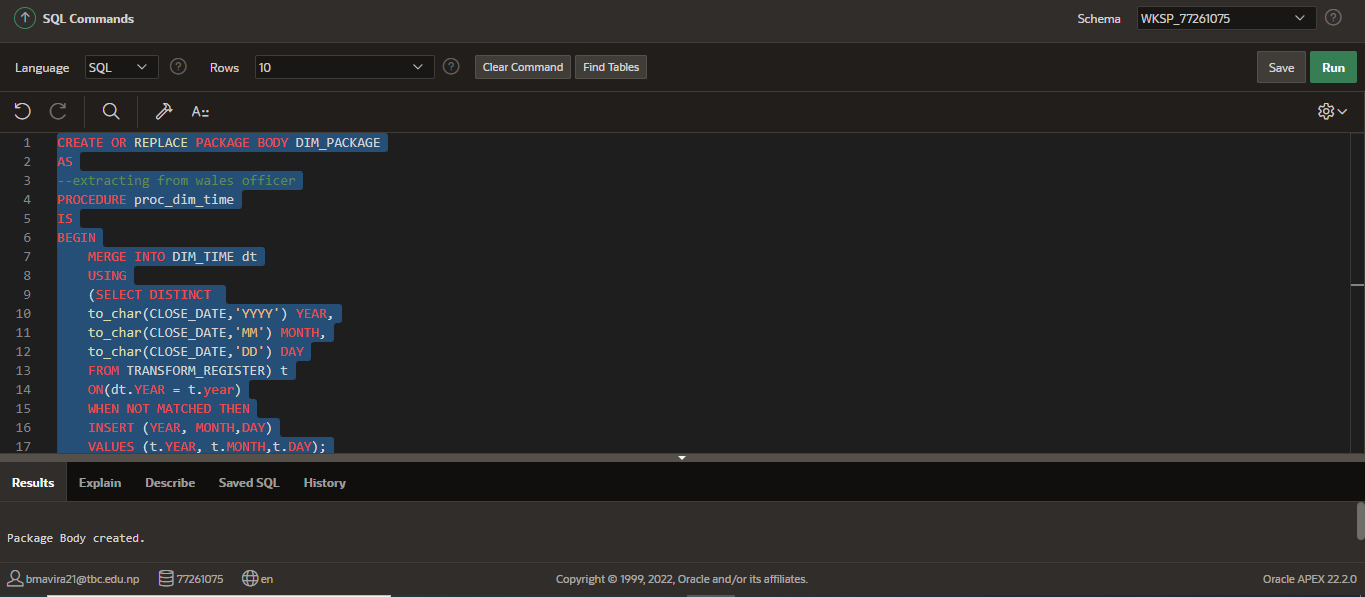


Figure: Making Procedure for DIM\_TIME

Package body declaration for dim time table.

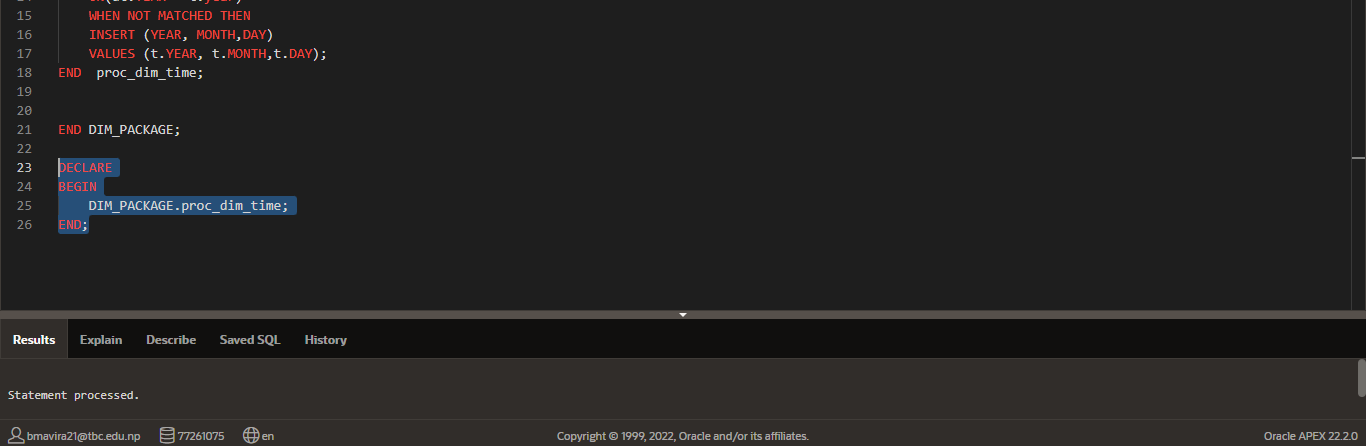


Figure: Declaring and running package

Calling the package for dim time table.

# 2. Inserting from data source PRCS

## a) Insert into stage table

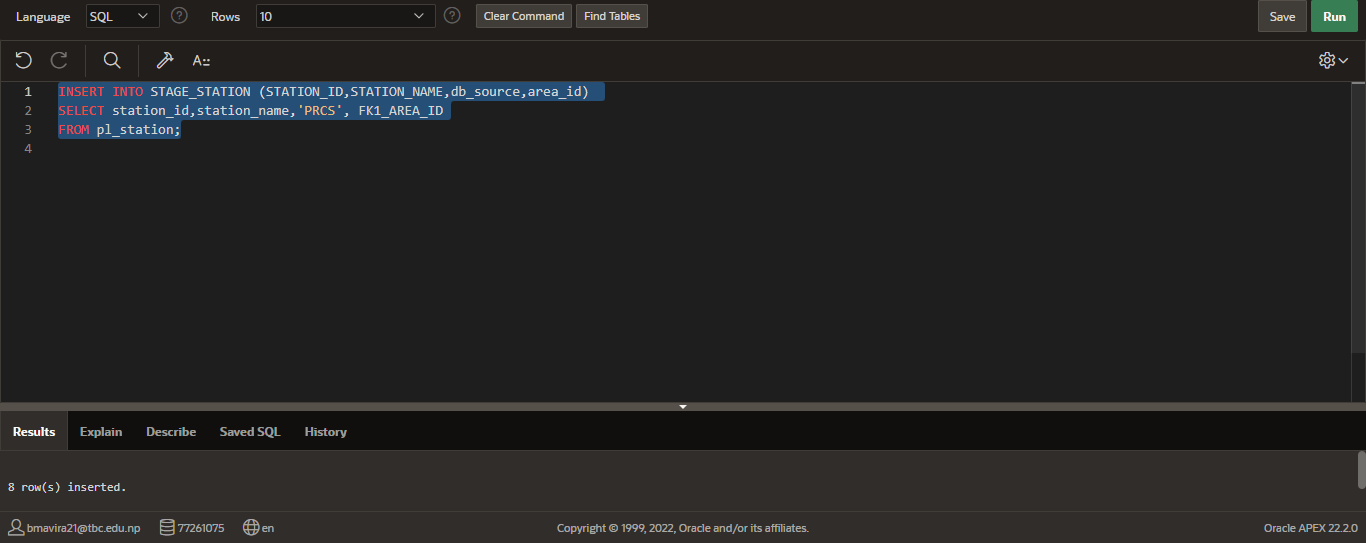


Figure: Inserting data from station table of prcs into staging table of station.

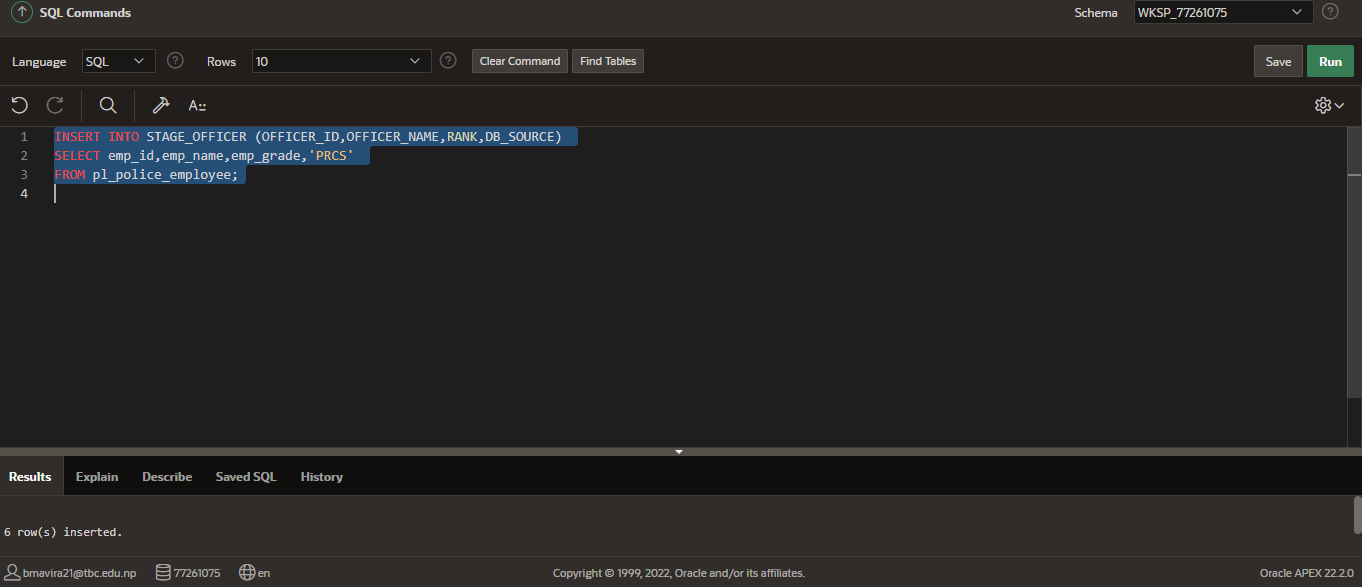


Figure: Inserting data from officer table of prcs into staging table of officer.

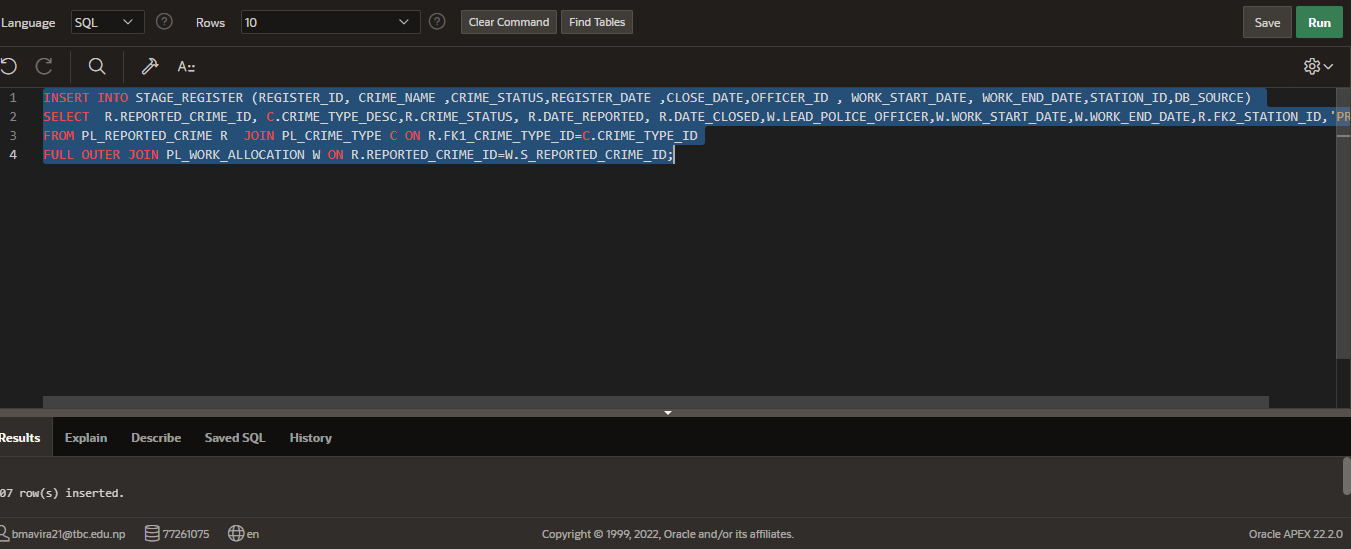


Figure: Inserting data from crime table of prcs into staging table of register.

b) Update in stage table

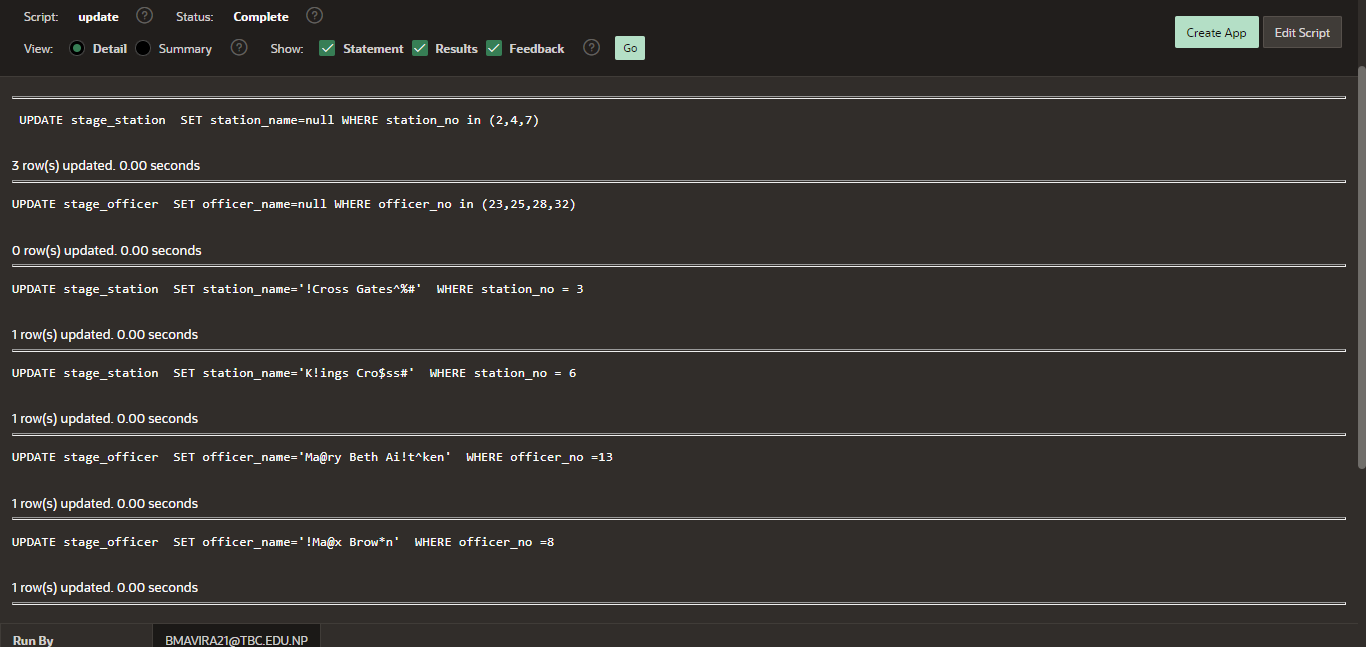


Figure: Updating data in Staging tables

Some datas are updated in stage\_register, stage\_station, stage\_officer respectively to make some bad data in order to insert into bad tables.

c) Inserting into dim tables

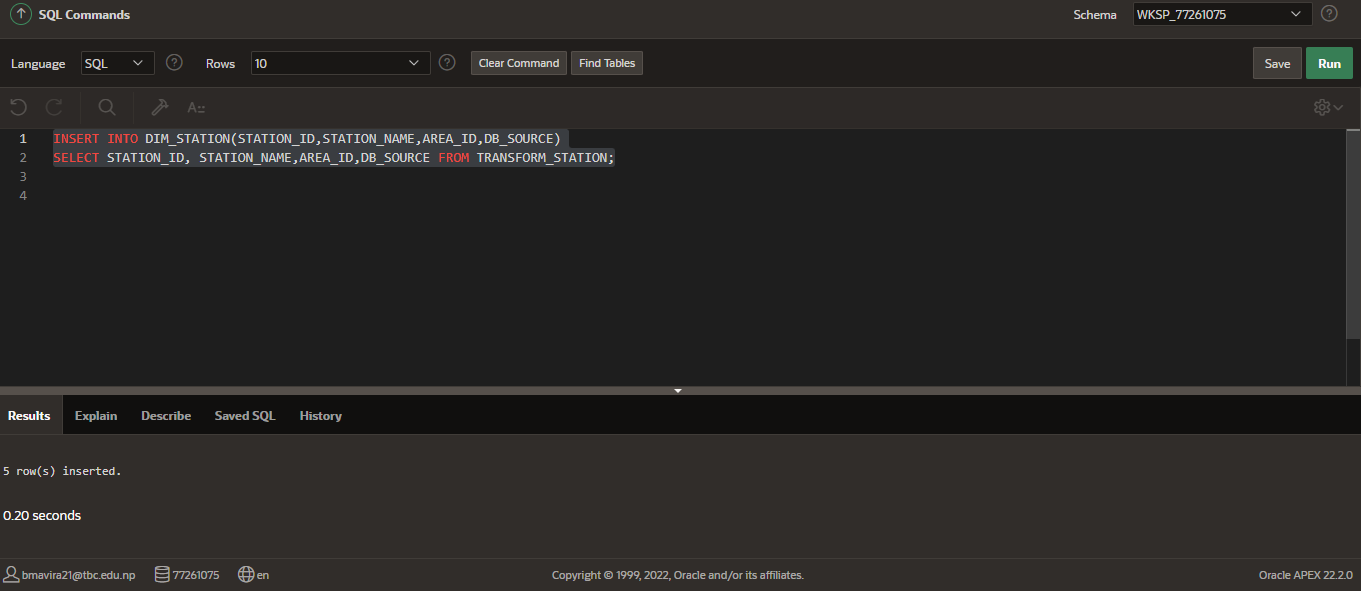


Figure: Inserting data into DIM\_STATION from TRANFORM\_STATION

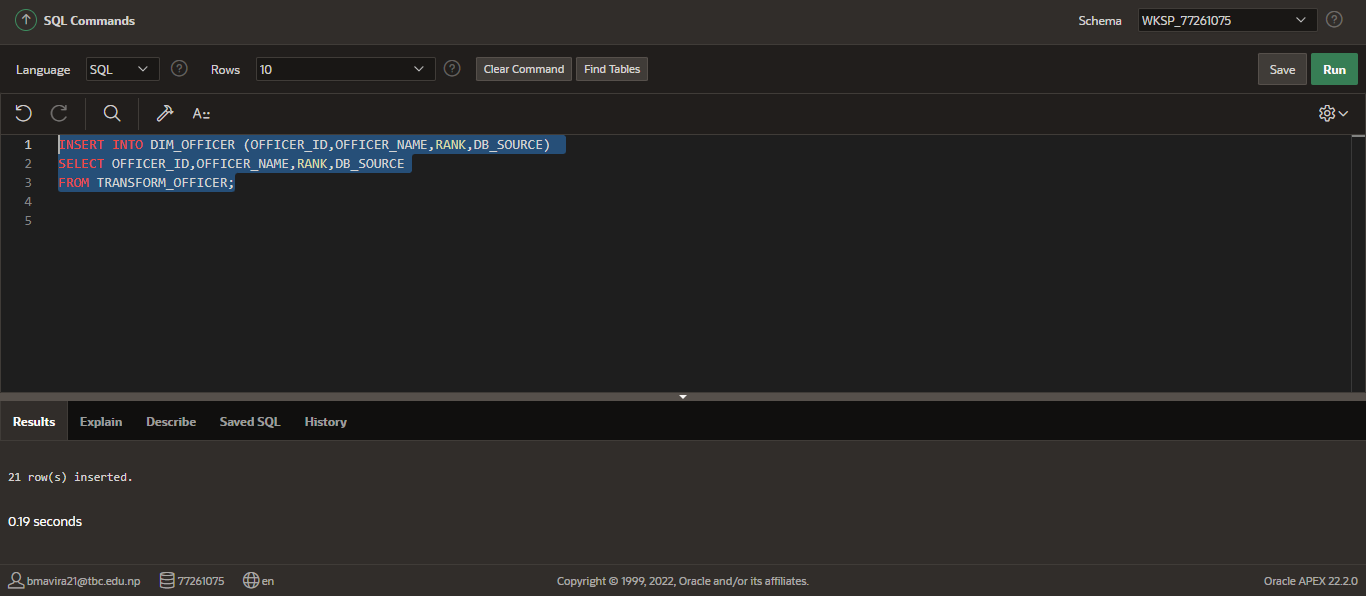


Figure: Inserting data into DIM\_OFFICER from TRANFORM\_OFFICER

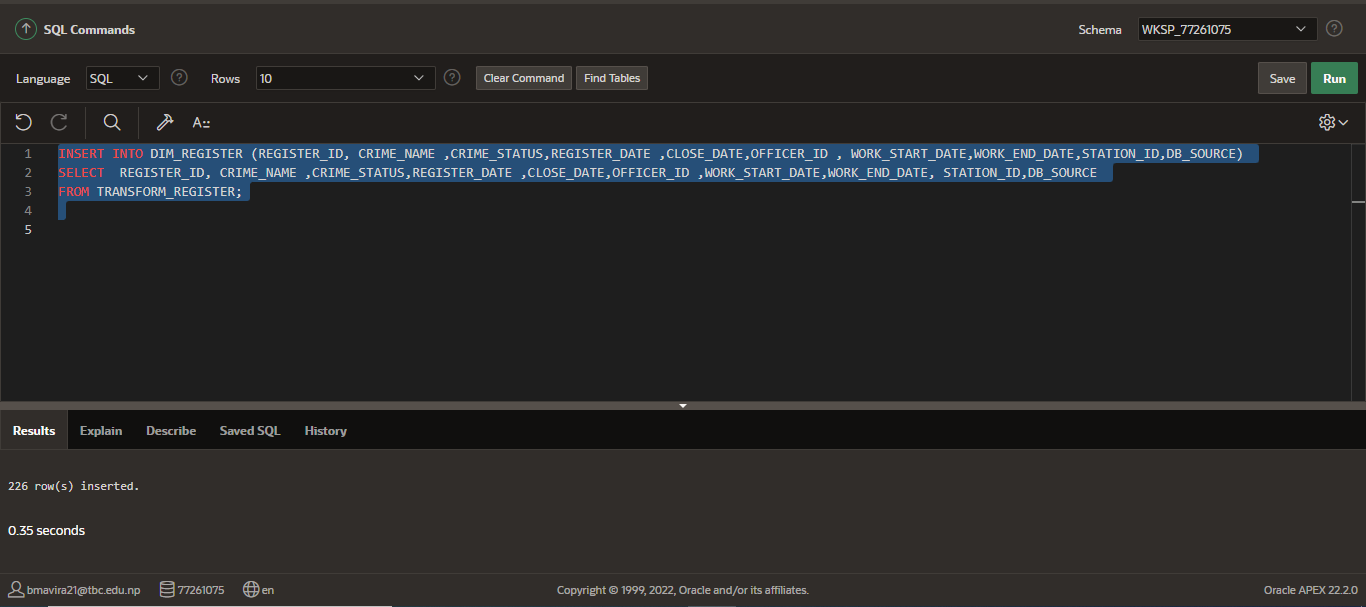


Figure: Inserting data into DIM\_REGISTER from TRANFORM\_REGISTER

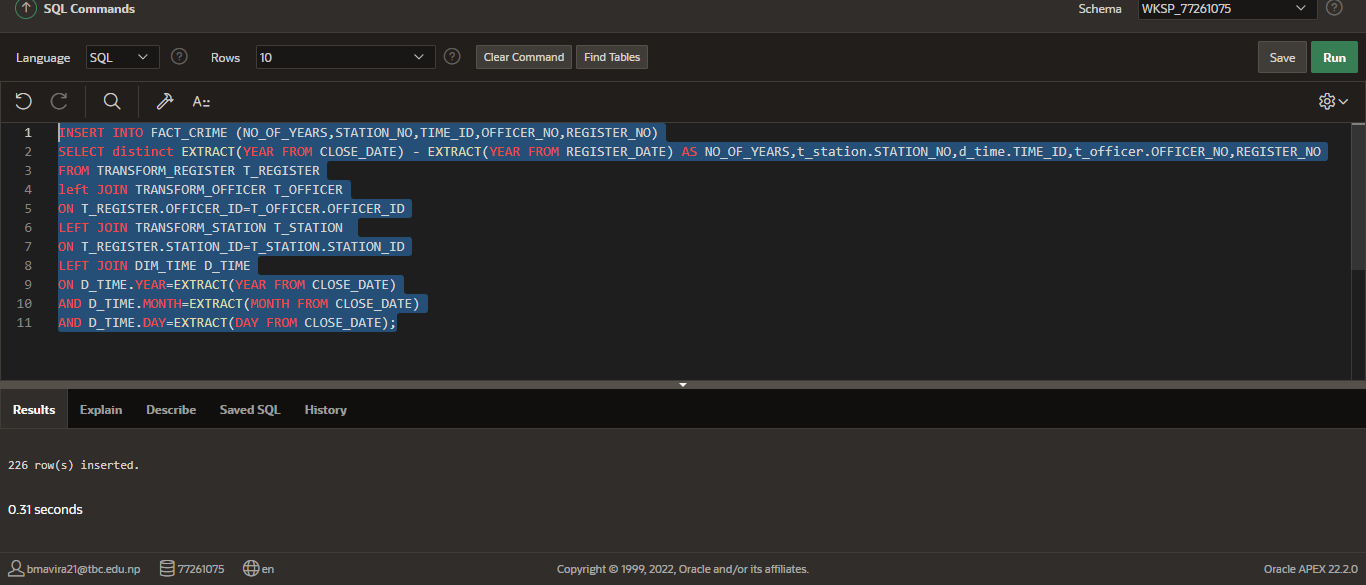


Figure: Inserting data into FACR\_CRIME

**SQL QUERIES**

**Packages**

1.

FOR STAGING TABLE

-----------------------------------------------

CREATE OR REPLACE PACKAGE STAGE\_PACKAGE

AS

PROCEDURE PRCS\_STAG\_OFFICER;

PROCEDURE PRCS\_STAG\_REGISTER ;

END STAGE\_PACKAGE;

CREATE OR REPLACE PACKAGE BODY STAGE\_PACKAGE

AS

--extracting from wales officer

PROCEDURE PRCS\_STAG\_OFFICER

IS

CURSOR CUR\_STAGE\_OFFICER

IS

SELECT officer\_id,first\_name || ' ' || middle\_name || ' ' ||last\_name as officer\_name,rank,'PS\_wales' AS DB\_SOURCE

FROM OFFICER WHERE officer\_id not in (SELECT OFFICER\_ID FROM STAGE\_OFFICER);

VOFFICER CUR\_STAGE\_OFFICER%ROWTYPE;

BEGIN

OPEN CUR\_STAGE\_OFFICER;

LOOP

FETCH CUR\_STAGE\_OFFICER INTO VOFFICER;

EXIT WHEN CUR\_STAGE\_OFFICER%NOTFOUND;

INSERT INTO STAGE\_OFFICER (OFFICER\_ID,OFFICER\_NAME,RANK,DB\_SOURCE) VALUES (VOFFICER.OFFICER\_ID,VOFFICER.OFFICER\_NAME,VOFFICER.RANK,VOFFICER.DB\_SOURCE);

END LOOP;

COMMIT;

CLOSE CUR\_STAGE\_OFFICER;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('RECORD NOT FOUND');

END PRCS\_STAG\_OFFICER;

PROCEDURE PRCS\_STAG\_REGISTER

IS

CURSOR CUR\_STAGE\_REGISTER

IS

SELECT crime\_id,crime\_name,crime\_status,reported\_date, closed\_date, police\_id,'PS\_wales' as DB\_SOURCE

FROM CRIME\_REGISTER;

VREGISTER CUR\_STAGE\_REGISTER%ROWTYPE;

BEGIN

OPEN CUR\_STAGE\_REGISTER;

LOOP

FETCH CUR\_STAGE\_REGISTER INTO VREGISTER;

EXIT WHEN CUR\_STAGE\_REGISTER%NOTFOUND;

INSERT INTO STAGE\_REGISTER (REGISTER\_ID,CRIME\_NAME,CRIME\_STATUS,REGISTER\_DATE,CLOSE\_DATE,OFFICER\_ID,WORK\_START\_DATE,WORK\_END\_DATE,STATION\_ID,DB\_SOURCE)

VALUES (VREGISTER.CRIME\_ID,VREGISTER.CRIME\_NAME,VREGISTER.CRIME\_STATUS,VREGISTER.REPORTED\_DATE,VREGISTER.CLOSED\_DATE,VREGISTER.POLICE\_ID,'','','',VREGISTER.DB\_SOURCE);

END LOOP;

COMMIT;

CLOSE CUR\_STAGE\_REGISTER;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('RECORD NOT FOUND');

END PRCS\_STAG\_REGISTER ;

END STAGE\_PACKAGE;

DECLARE

BEGIN

STAGE\_PACKAGE.PRCS\_STAG\_OFFICER;

STAGE\_PACKAGE.PRCS\_STAG\_REGISTER ;

END;

2.

BAD GOOD TABLE

CREATE OR REPLACE PACKAGE BADGOOD\_PACKAGE

AS

PROCEDURE proc\_filter\_officer;

PROCEDURE proc\_filter\_register;

PROCEDURE proc\_filter\_station;

END BADGOOD\_PACKAGE;

CREATE OR REPLACE PACKAGE BODY BADGOOD\_PACKAGE

AS

--extracting from wales officer

PROCEDURE proc\_filter\_officer

IS

CURSOR cur\_filter\_officer

IS

SELECT OFFICER\_ID,OFFICER\_NAME,RANK,DB\_SOURCE FROM STAGE\_OFFICER;

v\_officer\_id stage\_officer.officer\_id%type;

v\_officer\_name stage\_officer.officer\_name%type;

v\_rank stage\_officer.rank%type;

v\_db\_source stage\_officer.db\_source%type;

BEGIN

OPEN cur\_filter\_officer;

LOOP

FETCH cur\_filter\_officer

INTO

v\_officer\_id, v\_officer\_name,v\_rank, v\_db\_source;

EXIT WHEN cur\_filter\_officer%NOTFOUND;

IF (v\_officer\_name IS NULL OR REGEXP\_LIKE (v\_officer\_name, '[!%@\*^#$&]')) THEN

INSERT INTO BAD\_STAGE\_OFFICER (officer\_id, officer\_name, rank, error\_identification\_date, error\_resolution\_date, error\_description,error\_status,db\_source)

VALUES (v\_officer\_id,v\_officer\_name,v\_rank, SYSDATE,NULL,'NULL VALUE','NOT FIXED',v\_db\_source);

ELSE

INSERT INTO GOOD\_STAGE\_OFFICER (officer\_id, officer\_name,rank,db\_source)

VALUES (v\_officer\_id,v\_officer\_name, v\_rank,v\_db\_source);

END IF;

END LOOP;

COMMIT;

CLOSE cur\_filter\_officer;

EXCEPTION

WHEN no\_data\_found THEN

RAISE\_APPLICATION\_ERROR (-20001,'NO DATA FOUND');

END proc\_filter\_officer;

PROCEDURE proc\_filter\_register

IS

CURSOR cur\_filter\_register

IS

SELECT REGISTER\_ID,CRIME\_NAME,CRIME\_STATUS,REGISTER\_DATE,CLOSE\_DATE,OFFICER\_ID,WORK\_START\_DATE,WORK\_END\_DATE,STATION\_ID,DB\_SOURCE FROM STAGE\_REGISTER;

v\_register\_id stage\_register.register\_id%type;

v\_crime\_name stage\_register.crime\_name%type;

v\_crime\_status stage\_register.crime\_status%type;

v\_register\_date stage\_register.register\_date%type;

v\_close\_date stage\_register.close\_date%type;

v\_officer\_id stage\_register.officer\_id%type;

v\_station\_id stage\_register.station\_id%type;

v\_start\_date stage\_register.work\_start\_date%type;

v\_end\_date stage\_register.work\_end\_date%type;

v\_db\_source stage\_register.db\_source%type;

BEGIN

OPEN cur\_filter\_register;

LOOP

FETCH cur\_filter\_register

INTO

v\_register\_id, v\_crime\_name,v\_crime\_status,v\_register\_date,v\_close\_date,v\_officer\_id,v\_start\_date,v\_end\_date,v\_station\_id, v\_db\_source;

EXIT WHEN cur\_filter\_register%NOTFOUND;

IF (v\_crime\_name IS NULL OR v\_register\_date IS NULL OR v\_crime\_status IS NULL OR (v\_close\_date is NULL AND v\_crime\_status = 'Closed')OR (v\_officer\_id IS NOT NULL AND v\_crime\_status = 'Closed') OR (v\_officer\_id IS NOT NULL AND v\_start\_date IS NULL)OR v\_officer\_id IS NULL OR v\_station\_id IS NULL OR v\_register\_date>v\_close\_date OR v\_start\_date>v\_end\_date) THEN

INSERT INTO BAD\_STAGE\_REGISTER (register\_id, crime\_name, crime\_status, register\_date,close\_date, officer\_id, work\_start\_date, work\_end\_date, station\_id, error\_identification\_date, error\_resolution\_date, error\_description,error\_status,db\_source)

VALUES (v\_register\_id,v\_crime\_name, v\_crime\_status, v\_register\_date,v\_close\_date,v\_officer\_id,v\_start\_date,v\_end\_date,v\_station\_id, SYSDATE,NULL,'NULL VALUE','NOT FIXED',v\_db\_source);

ELSE

INSERT INTO GOOD\_STAGE\_REGISTER (register\_id, crime\_name, crime\_status, register\_date,close\_date, officer\_id, work\_start\_date,work\_end\_date, station\_id ,db\_source)

VALUES (v\_register\_id,v\_crime\_name, v\_crime\_status, v\_register\_date,v\_close\_date,v\_officer\_id,v\_start\_date,v\_end\_date,v\_station\_id, v\_db\_source);

END IF;

END LOOP;

COMMIT;

CLOSE cur\_filter\_register;

EXCEPTION

WHEN no\_data\_found THEN

RAISE\_APPLICATION\_ERROR (-20001,'NO DATA FOUND');

END proc\_filter\_register;

PROCEDURE proc\_filter\_station

IS

CURSOR cur\_filter\_station

IS

SELECT STATION\_ID,STATION\_NAME,AREA\_ID,DB\_SOURCE FROM STAGE\_STATION;

v\_station\_id stage\_station.station\_id%type;

v\_station\_name stage\_station.station\_name%type;

v\_area\_id stage\_station.area\_id%type;

v\_db\_source stage\_station.db\_source%type;

BEGIN

OPEN cur\_filter\_station;

LOOP

FETCH cur\_filter\_station

INTO

v\_station\_id, v\_station\_name,v\_area\_id, v\_db\_source;

EXIT WHEN cur\_filter\_station%NOTFOUND;

IF (v\_station\_name IS NULL OR REGEXP\_LIKE (v\_station\_name, '[!%^#$&]')) THEN

INSERT INTO BAD\_STAGE\_STATION (station\_id, station\_name,area\_id, error\_identification\_date, error\_resolution\_date, error\_description,error\_status,db\_source)

VALUES (v\_station\_id,v\_station\_name,v\_area\_id, SYSDATE,NULL,'NULL VALUE','NOT FIXED',v\_db\_source);

ELSE

INSERT INTO GOOD\_STAGE\_STATION (station\_id, station\_name,area\_id,db\_source)

VALUES (v\_station\_id,v\_station\_name,v\_area\_id, v\_db\_source);

END IF;

END LOOP;

COMMIT;

CLOSE cur\_filter\_station;

EXCEPTION

WHEN no\_data\_found THEN

RAISE\_APPLICATION\_ERROR (-20001,'NO DATA FOUND');

END proc\_filter\_station;

END BADGOOD\_PACKAGE;

DECLARE

BEGIN

BADGOOD\_PACKAGE. proc\_filter\_officer;

BADGOOD\_PACKAGE.proc\_filter\_register;

BADGOOD\_PACKAGE. proc\_filter\_station;

END;

3.

RECLEAN

CREATE OR REPLACE PACKAGE RECLEAN\_PACKAGE

AS

PROCEDURE clean\_station;

PROCEDURE clean\_register;

PROCEDURE clean\_officer;

END RECLEAN\_PACKAGE;

CREATE OR REPLACE PACKAGE BODY RECLEAN\_PACKAGE

AS

--extracting from wales officer

PROCEDURE clean\_station IS

CURSOR cur\_station

IS

SELECT \* FROM bad\_stage\_station ;

BEGIN

FOR v in cur\_station

LOOP

IF (v.station\_name is null)

THEN

UPDATE bad\_stage\_station SET STATION\_NAME ='n/a' ,ERROR\_RESOLUTION\_DATE=sysdate, ERROR\_STATUS='FIXED'

WHERE STATION\_NAME IS NULL;

ELSE

UPDATE bad\_stage\_station SET STATION\_NAME = regexp\_replace(v.station\_name,'[!%^#$&]'),

ERROR\_RESOLUTION\_DATE=CURRENT\_TIMESTAMP, ERROR\_STATUS='FIXED'

WHERE REGEXP\_LIKE(STATION\_NAME,'[!%^#$&]');

END IF;

END LOOP;

END clean\_station;

PROCEDURE clean\_register IS

CURSOR cur\_register

IS

SELECT \* FROM bad\_stage\_register ;

BEGIN

FOR v in cur\_register

LOOP

IF (v.crime\_name is null)

THEN

UPDATE bad\_stage\_register SET CRIME\_NAME ='n/a' ,ERROR\_RESOLUTION\_DATE=sysdate, ERROR\_STATUS='FIXED'

WHERE CRIME\_NAME IS NULL;

ELSIF(v.officer\_id is null)

THEN

UPDATE bad\_stage\_register SET officer\_id =0 ,ERROR\_RESOLUTION\_DATE=sysdate, ERROR\_STATUS='FIXED'

where officer\_id is null;

ELSIF(v.station\_id is null)

THEN

UPDATE bad\_stage\_register SET station\_id =0 ,ERROR\_RESOLUTION\_DATE=sysdate, ERROR\_STATUS='FIXED'

where station\_id is null;

ELSIF(v.crime\_status='CLOSED')

THEN

UPDATE bad\_stage\_register SET CLOSE\_DATE=SYSDATE,

ERROR\_RESOLUTION\_DATE=CURRENT\_TIMESTAMP, ERROR\_STATUS='FIXED'

WHERE CRIME\_STATUS='CLOSED';

ELSIF(v.work\_start\_date>v.work\_end\_date)

THEN

UPDATE bad\_stage\_register SET WORK\_END\_DATE=sysdate,

ERROR\_RESOLUTION\_DATE=CURRENT\_TIMESTAMP, ERROR\_STATUS='FIXED'

WHERE WORK\_START\_DATE>WORK\_END\_DATE;

ELSE

UPDATE bad\_stage\_register SET CLOSE\_DATE=SYSDATE,

ERROR\_RESOLUTION\_DATE=CURRENT\_TIMESTAMP, ERROR\_STATUS='FIXED'

WHERE REGISTER\_DATE>CLOSE\_DATE;

end if;

END LOOP;

END clean\_register;

PROCEDURE clean\_officer IS

CURSOR cur\_officer

IS

SELECT \* FROM bad\_stage\_officer;

BEGIN

FOR v in cur\_officer

LOOP

IF (v.officer\_name is null)

THEN

UPDATE bad\_stage\_officer SET OFFICER\_NAME ='n/a' ,ERROR\_RESOLUTION\_DATE=sysdate, ERROR\_STATUS='FIXED'

WHERE OFFICER\_NAME IS NULL;

ELSE

UPDATE bad\_stage\_officer SET OFFICER\_NAME = regexp\_replace(v.officer\_name,'[!%@\*^#$&]'),

ERROR\_RESOLUTION\_DATE=CURRENT\_TIMESTAMP, ERROR\_STATUS='FIXED'

WHERE REGEXP\_LIKE(OFFICER\_NAME,'[!%@\*^#$&]');

END IF;

END LOOP;

END clean\_officer;

END RECLEAN\_PACKAGE;

DECLARE

BEGIN

RECLEAN\_PACKAGE. clean\_station;

RECLEAN\_PACKAGE.clean\_register;

RECLEAN\_PACKAGE. clean\_officer;

END;

4.

TRANSFORM

------------------------

CREATE OR REPLACE PACKAGE TRANSFORM\_PACKAGE

AS

PROCEDURE PRCS\_TRANS\_STATION;

PROCEDURE PRCS\_TRANS\_OFFICER;

PROCEDURE PRCS\_TRANS\_REGISTER;

END TRANSFORM\_PACKAGE;

CREATE OR REPLACE PACKAGE BODY TRANSFORM\_PACKAGE

AS

--extracting from wales officer

PROCEDURE PRCS\_TRANS\_STATION

IS

CURSOR CUR\_TRANS\_STATION

IS

SELECT station\_id, station\_name,area\_id,DB\_SOURCE

FROM GOOD\_STAGE\_STATION WHERE station\_id not in (SELECT station\_ID FROM TRANSFORM\_OFFICER);

VSTATION CUR\_TRANS\_STATION%ROWTYPE;

BEGIN

OPEN CUR\_TRANS\_STATION;

LOOP

FETCH CUR\_TRANS\_STATION INTO VSTATION;

EXIT WHEN CUR\_TRANS\_STATION%NOTFOUND;

INSERT INTO TRANSFORM\_STATION (STATION\_ID,STATION\_NAME,AREA\_ID,DB\_SOURCE) VALUES (VSTATION.STATION\_ID,VSTATION.STATION\_NAME,VSTATION.AREA\_ID,VSTATION.DB\_SOURCE);

END LOOP;

COMMIT;

CLOSE CUR\_TRANS\_STATION;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('RECORD NOT FOUND');

END PRCS\_TRANS\_STATION;

PROCEDURE PRCS\_TRANS\_OFFICER

IS

CURSOR CUR\_TRANS\_OFFICER

IS

SELECT officer\_id, officer\_name,RANK, DB\_SOURCE

FROM GOOD\_STAGE\_OFFICER WHERE officer\_id not in (SELECT OFFICER\_ID FROM TRANSFORM\_OFFICER);

VOFFICER CUR\_TRANS\_OFFICER%ROWTYPE;

BEGIN

OPEN CUR\_TRANS\_OFFICER;

LOOP

FETCH CUR\_TRANS\_OFFICER INTO VOFFICER;

EXIT WHEN CUR\_TRANS\_OFFICER%NOTFOUND;

INSERT INTO TRANSFORM\_OFFICER (OFFICER\_ID,OFFICER\_NAME,RANK,DB\_SOURCE) VALUES (VOFFICER.OFFICER\_ID,VOFFICER.OFFICER\_NAME,VOFFICER.RANK,VOFFICER.DB\_SOURCE);

END LOOP;

COMMIT;

CLOSE CUR\_TRANS\_OFFICER;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('RECORD NOT FOUND');

END PRCS\_TRANS\_OFFICER;

PROCEDURE PRCS\_TRANS\_REGISTER

IS

CURSOR CUR\_TRANS\_REGISTER

IS

SELECT register\_id,crime\_name,crime\_status,register\_date, close\_date, officer\_id,work\_start\_date,work\_end\_date,station\_id,DB\_SOURCE

FROM GOOD\_STAGE\_REGISTER;

VREGISTER CUR\_TRANS\_REGISTER%ROWTYPE;

BEGIN

OPEN CUR\_TRANS\_REGISTER;

LOOP

FETCH CUR\_TRANS\_REGISTER INTO VREGISTER;

EXIT WHEN CUR\_TRANS\_REGISTER%NOTFOUND;

INSERT INTO TRANSFORM\_REGISTER (REGISTER\_ID,CRIME\_NAME,CRIME\_STATUS,REGISTER\_DATE,CLOSE\_DATE,OFFICER\_ID,WORK\_START\_DATE,WORK\_END\_DATE,STATION\_ID,DB\_SOURCE)

VALUES (VREGISTER.REGISTER\_ID,VREGISTER.CRIME\_NAME,VREGISTER.CRIME\_STATUS,VREGISTER.REGISTER\_DATE,VREGISTER.CLOSE\_DATE,VREGISTER.OFFICER\_ID,VREGISTER.WORK\_START\_DATE,VREGISTER.WORK\_END\_DATE,VREGISTER.STATION\_ID,VREGISTER.DB\_SOURCE);

END LOOP;

COMMIT;

CLOSE CUR\_TRANS\_REGISTER;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('RECORD NOT FOUND');

END PRCS\_TRANS\_REGISTER;

END TRANSFORM\_PACKAGE;

DECLARE

BEGIN

TRANSFORM\_PACKAGE.PRCS\_TRANS\_STATION;

TRANSFORM\_PACKAGE.PRCS\_TRANS\_OFFICER;

TRANSFORM\_PACKAGE. PRCS\_TRANS\_REGISTER;

END;

-----------------

Fact

CREATE OR REPLACE PACKAGE DIM\_PACKAGE

AS

PROCEDURE proc\_dim\_time;

END DIM\_PACKAGE;

CREATE OR REPLACE PACKAGE BODY DIM\_PACKAGE

AS

--extracting from wales officer

PROCEDURE proc\_dim\_time

IS

BEGIN

MERGE INTO DIM\_TIME dt

USING

(SELECT DISTINCT

to\_char(CLOSE\_DATE,'YYYY') YEAR,

to\_char(CLOSE\_DATE,'MM') MONTH,

to\_char(CLOSE\_DATE,'DD') DAY

FROM TRANSFORM\_REGISTER) t

ON(dt.YEAR = t.year)

WHEN NOT MATCHED THEN

INSERT (YEAR, MONTH,DAY)

VALUES (t.YEAR, t.MONTH,t.DAY);

END proc\_dim\_time;

END DIM\_PACKAGE;

DECLARE

BEGIN

DIM\_PACKAGE.proc\_dim\_time;

END;

**Insert Queries**

1ST STAGE INSERTING

INSERT INTO STAGE\_STATION (STATION\_ID,STATION\_NAME,db\_source,area\_id)

SELECT station\_id,station\_name,'PRCS', FK1\_AREA\_ID

FROM pl\_station;

INSERT INTO STAGE\_OFFICER (OFFICER\_ID,OFFICER\_NAME,RANK,DB\_SOURCE)

SELECT emp\_id,emp\_name,emp\_grade,'PRCS'

FROM pl\_police\_employee;

INSERT INTO STAGE\_REGISTER (REGISTER\_ID, CRIME\_NAME ,CRIME\_STATUS,REGISTER\_DATE ,CLOSE\_DATE,OFFICER\_ID , WORK\_START\_DATE, WORK\_END\_DATE,STATION\_ID,DB\_SOURCE)

SELECT R.REPORTED\_CRIME\_ID, C.CRIME\_TYPE\_DESC,R.CRIME\_STATUS, R.DATE\_REPORTED, R.DATE\_CLOSED,W.LEAD\_POLICE\_OFFICER,W.WORK\_START\_DATE,W.WORK\_END\_DATE,R.FK2\_STATION\_ID,'PRCS'

FROM PL\_REPORTED\_CRIME R JOIN PL\_CRIME\_TYPE C ON R.FK1\_CRIME\_TYPE\_ID=C.CRIME\_TYPE\_ID

FULL OUTER JOIN PL\_WORK\_ALLOCATION W ON R.REPORTED\_CRIME\_ID=W.S\_REPORTED\_CRIME\_ID;

**Inserting into dim table and fact table**

INSERT INTO DIM\_STATION(STATION\_ID,STATION\_NAME,AREA\_ID,DB\_SOURCE)

SELECT STATION\_ID, STATION\_NAME,AREA\_ID,DB\_SOURCE FROM TRANSFORM\_STATION;

3.

INSERT INTO DIM\_OFFICER (OFFICER\_ID,OFFICER\_NAME,RANK,DB\_SOURCE)

SELECT OFFICER\_ID,OFFICER\_NAME,RANK,DB\_SOURCE

FROM TRANSFORM\_OFFICER;

4.

INSERT INTO DIM\_REGISTER (REGISTER\_ID, CRIME\_NAME ,CRIME\_STATUS,REGISTER\_DATE ,CLOSE\_DATE,OFFICER\_ID , WORK\_START\_DATE,WORK\_END\_DATE,STATION\_ID,DB\_SOURCE)

SELECT REGISTER\_ID, CRIME\_NAME ,CRIME\_STATUS,REGISTER\_DATE ,CLOSE\_DATE,OFFICER\_ID ,WORK\_START\_DATE,WORK\_END\_DATE, STATION\_ID,DB\_SOURCE

FROM TRANSFORM\_REGISTER;

5.

INSERT INTO FACT\_CRIME (NO\_OF\_YEARS,STATION\_NO,TIME\_ID,OFFICER\_NO,REGISTER\_NO)

SELECT distinct EXTRACT(YEAR FROM CLOSE\_DATE) - EXTRACT(YEAR FROM REGISTER\_DATE) AS NO\_OF\_YEARS,t\_station.STATION\_NO,d\_time.TIME\_ID,t\_officer.OFFICER\_NO,REGISTER\_NO

FROM TRANSFORM\_REGISTER T\_REGISTER

left JOIN TRANSFORM\_OFFICER T\_OFFICER

ON T\_REGISTER.OFFICER\_ID=T\_OFFICER.OFFICER\_ID

LEFT JOIN TRANSFORM\_STATION T\_STATION

ON T\_REGISTER.STATION\_ID=T\_STATION.STATION\_ID

LEFT JOIN DIM\_TIME D\_TIME

ON D\_TIME.YEAR=EXTRACT(YEAR FROM CLOSE\_DATE)

AND D\_TIME.MONTH=EXTRACT(MONTH FROM CLOSE\_DATE)

AND D\_TIME.DAY=EXTRACT(DAY FROM CLOSE\_DATE)