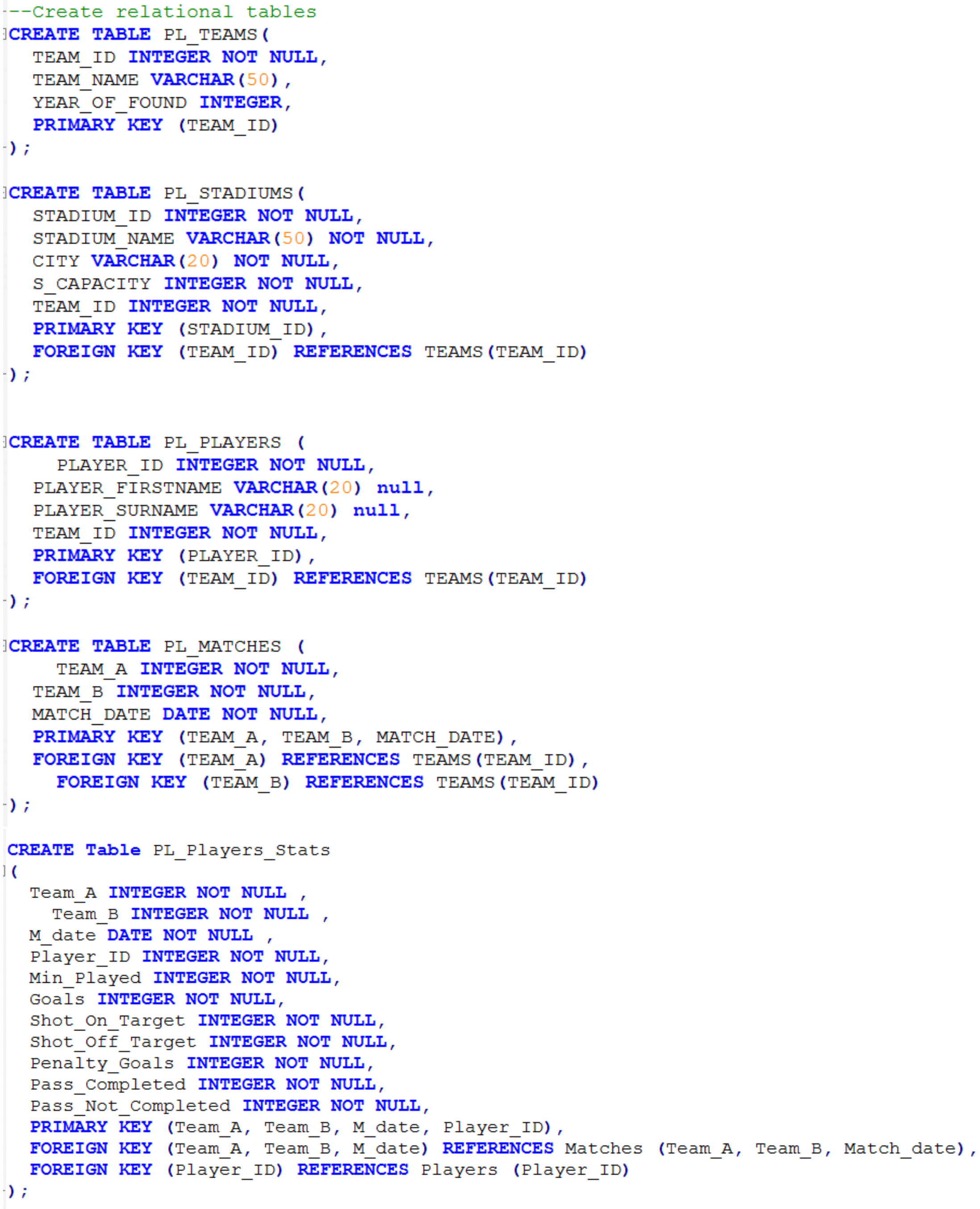
DATABASES ASSIGNMENT

Student name: Ronan Dillon

Student number: C12335251

PART A

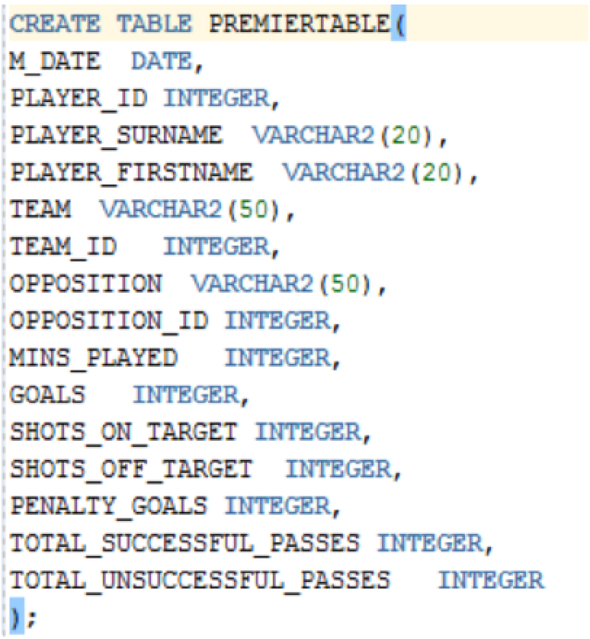
For part A I used 5 create table statements to make up the relational model. I entered all their fields and gave the tables primary and foreign keys with the correct corresponding tables.



```
--Create relational tables
CREATE TABLE PL_TEAMS(
  TEAM_ID INTEGER NOT NULL,
  TEAM_NAME VARCHAR(50),
  YEAR_OF_FOUND INTEGER,
  PRIMARY KEY (TEAM_ID)
);

CREATE TABLE PL_STADIUMS(
  STADIUM_ID INTEGER NOT NULL,
  STADIUM_NAME VARCHAR(50) NOT NULL,
  CITY VARCHAR(20) NOT NULL,
  S_CAPACITY INTEGER NOT NULL,
  TEAM_ID INTEGER NOT NULL,
  PRIMARY KEY (STADIUM_ID),
  FOREIGN KEY (TEAM_ID) REFERENCES TEAMS(TEAM_ID)
);


CREATE TABLE PL_PLAYERS (
    PLAYER_ID INTEGER NOT NULL,
  PLAYER_FIRSTNAME VARCHAR(20) null,
  PLAYER_SURNAME VARCHAR(20) null,
  TEAM_ID INTEGER NOT NULL,
  PRIMARY KEY (PLAYER_ID),
  FOREIGN KEY (TEAM_ID) REFERENCES TEAMS(TEAM_ID)
);

CREATE TABLE PL_MATCHES (
    TEAM_A INTEGER NOT NULL,
  TEAM_B INTEGER NOT NULL,
  MATCH_DATE DATE NOT NULL,
  PRIMARY KEY (TEAM_A, TEAM_B, MATCH_DATE),
  FOREIGN KEY (TEAM_A) REFERENCES TEAMS(TEAM_ID),
    FOREIGN KEY (TEAM_B) REFERENCES TEAMS(TEAM_ID)
);

CREATE Table PL_Players_Stats
(
  Team_A INTEGER NOT NULL ,
    Team_B INTEGER NOT NULL ,
  M_date DATE NOT NULL ,
  Player_ID INTEGER NOT NULL,
  Min_Played INTEGER NOT NULL,
  Goals INTEGER NOT NULL,
  Shot_On_Target INTEGER NOT NULL,
  Shot_Off_Target INTEGER NOT NULL,
  Penalty_Goals INTEGER NOT NULL,
  Pass_Completed INTEGER NOT NULL,
  Pass_Not_Completed INTEGER NOT NULL,
  PRIMARY KEY (Team_A, Team_B, M_date, Player_ID),
  FOREIGN KEY (Team_A, Team_B, M_date) REFERENCES Matches (Team_A, Team_B, Match_date),
  FOREIGN KEY (Player_ID) REFERENCES Players (Player_ID)
);
```

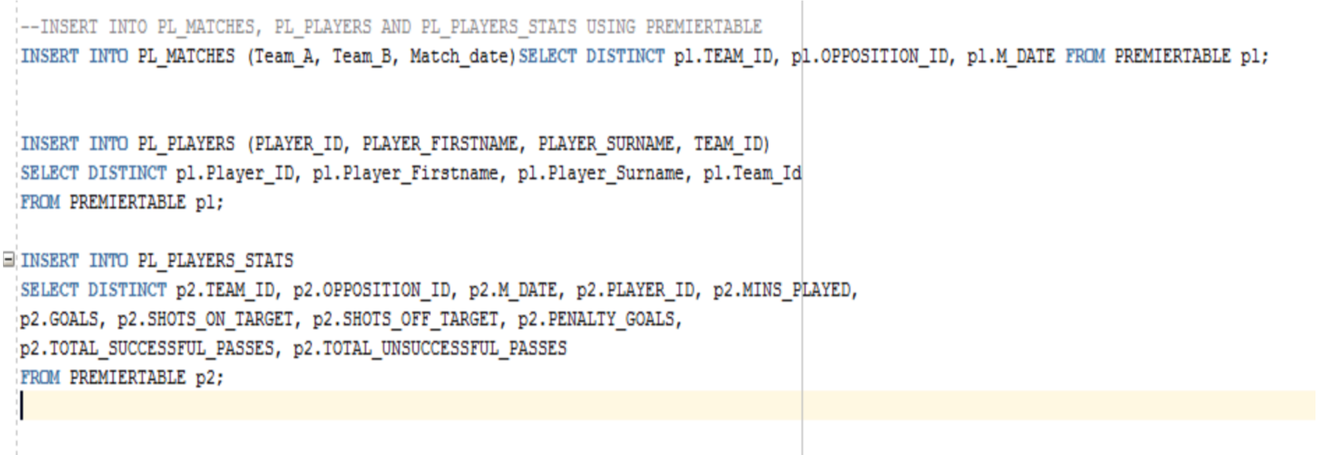
PART B

I started part B by creating a denormalized table. I then imported the csv file premier.csv into the denormalized table.



```
CREATE TABLE PREMIERTABLE(
M_DATE  DATE,
PLAYER_ID INTEGER,
PLAYER_SURNAME  VARCHAR2(20),
PLAYER_FIRSTNAME  VARCHAR2(20),
TEAM  VARCHAR2(50),
TEAM_ID   INTEGER,
OPPOSITION  VARCHAR2(50),
OPPOSITION_ID INTEGER,
MINS_PLAYED   INTEGER,
GOALS    INTEGER,
SHOTS_ON_TARGET INTEGER,
SHOTS_OFF_TARGET  INTEGER,
PENALTY_GOALS INTEGER,
TOTAL_SUCCESSFUL_PASSES INTEGER,
TOTAL_UNSUCCESSFUL_PASSES   INTEGER
);
```

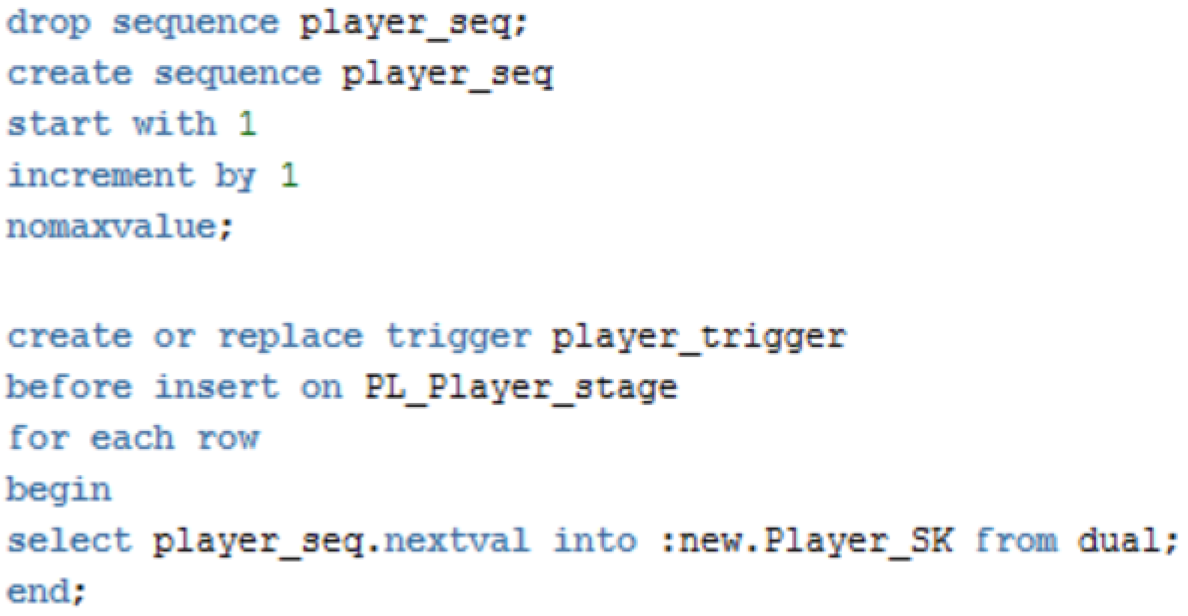
I then used the statements in insert.sql to insert data into the PL_TEAMS and PL_STADIUMS tables. After that I wrote insert statements to insert into the other three tables using data in PREMIERTABLE.



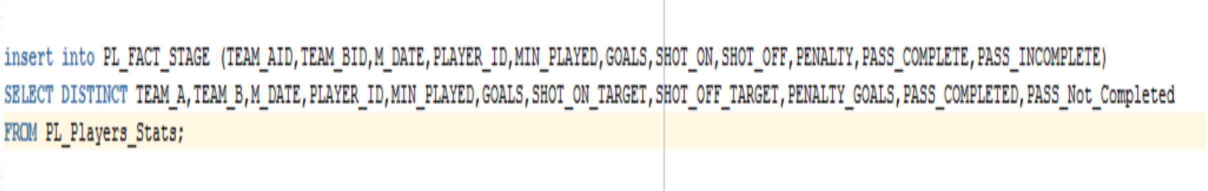
```
--INSERT INTO PL_MATCHES, PL_PLAYERS AND PL_PLAYERS_STATS USING PREMIERTABLE
INSERT INTO PL_MATCHES (Team_A, Team_B, Match_date)SELECT DISTINCT p1.TEAM_ID, p1.OPPOSITION_ID, p1.M_DATE FROM PREMIERTABLE p1;


INSERT INTO PL_PLAYERS (PLAYER_ID, PLAYER_FIRSTNAME, PLAYER_SURNAME, TEAM_ID)
SELECT DISTINCT p1.Player_ID, p1.Player_Firstname, p1.Player_Surname, p1.Team_Id
FROM PREMIERTABLE p1;

INSERT INTO PL_PLAYERS_STATS
SELECT DISTINCT p2.TEAM_ID, p2.OPPOSITION_ID, p2.M_DATE, p2.PLAYER_ID, p2.MINS_PLAYED,
p2.GOALS, p2.SHOTS_ON_TARGET, p2.SHOTS_OFF_TARGET, p2.PENALTY_GOALS,
p2.TOTAL_SUCCESSFUL_PASSES, p2.TOTAL_UNSUCCESSFUL_PASSES
FROM PREMIERTABLE p2;
```

PART C

For part C I created stage tables, using the data in my relational model I moved data from the relational tables into the stage tables. Using sequences and triggers a new and unique SK was given to each stage table. The example below of the player trigger shows that before inserting into the PL_PLAYER_STAGE table the next value in the player sequence is added the player_sk. After adding to the stage table a procedure is called to update the SKs in the fact stage table so that they match up with the SKs in the other stage tables.

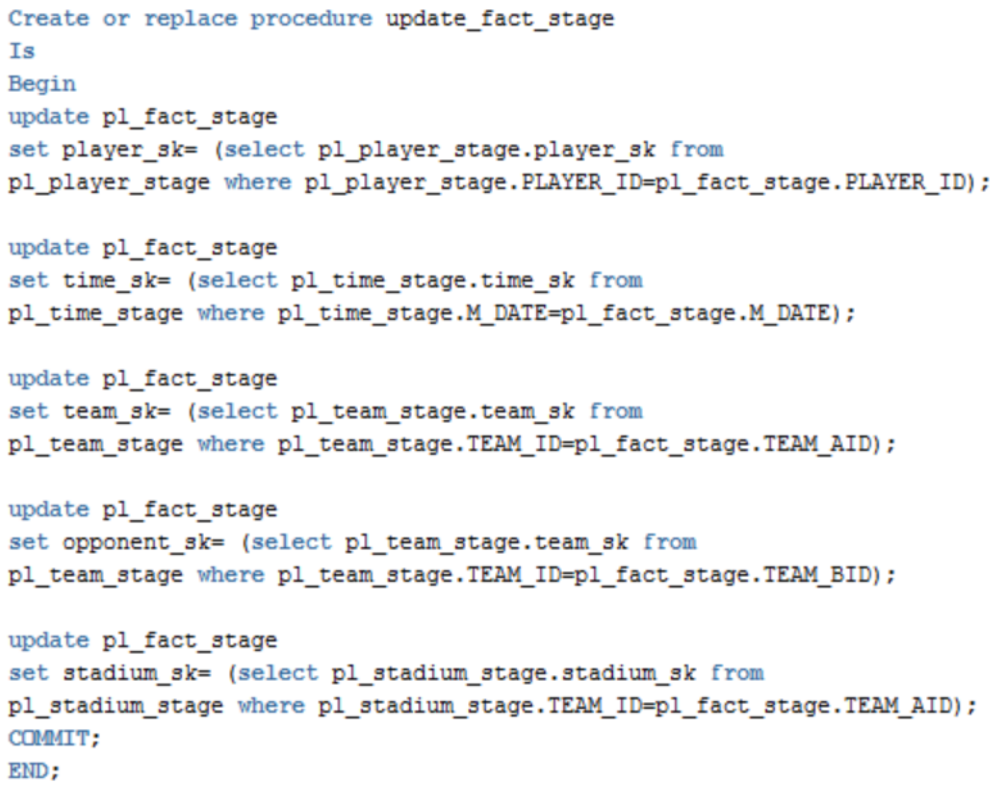


```
drop sequence player_seq;
create sequence player_seq
start with 1
increment by 1
nomaxvalue;

create or replace trigger player_trigger
before insert on PL_Player_stage
for each row
begin
select player_seq.nextval into :new.Player_SK from dual;
end;
```



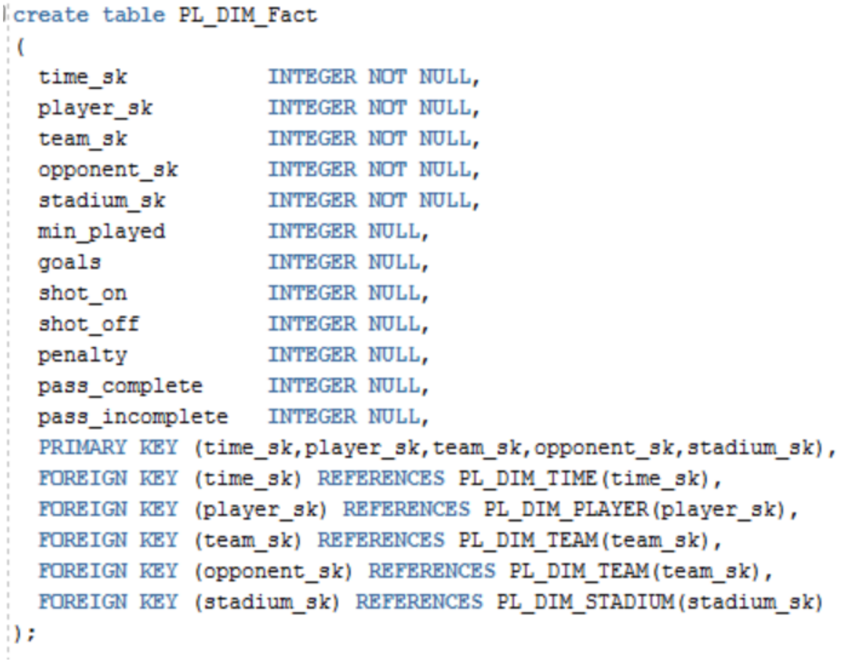
```
insert into PL_FACT_STAGE (TEAM_AID,TEAM_BID,M_DATE,PLAYER_ID,MIN_PLAYED,GOALS,SHOT_ON,SHOT_OFF,PENALTY,PASS_COMPLETE,PASS_INCOMPLETE)
SELECT DISTINCT TEAM_A,TEAM_B,M_DATE,PLAYER_ID,MIN_PLAYED,GOALS,SHOT_ON_TARGET,SHOT_OFF_TARGET,PENALTY_GOALS,PASS_COMPLETED,PASS_Not_Completed
FROM PL_Players_Stats;
```

Although none of the dimensional tables contain team A and B IDS, match date or player ID, I have put them in the stage tables to use for adding the SKs to the fact stage table. I use the values in the fact stage to match up with the corresponding stage tables.

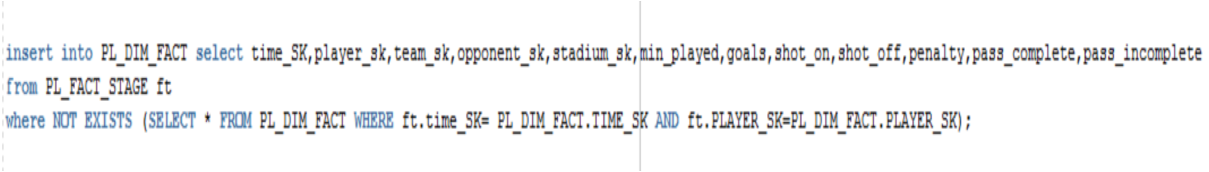


```
Create or replace procedure update_fact_stage
Is
Begin
update pl_fact_stage
set player_sk= (select pl_player_stage.player_sk from
pl_player_stage where pl_player_stage.PLAYER_ID=pl_fact_stage.PLAYER_ID);

update pl_fact_stage
set time_sk= (select pl_time_stage.time_sk from
pl_time_stage where pl_time_stage.M_DATE=pl_fact_stage.M_DATE);

update pl_fact_stage
set team_sk= (select pl_team_stage.team_sk from
pl_team_stage where pl_team_stage.TEAM_ID=pl_fact_stage.TEAM_AID);

update pl_fact_stage
set opponent_sk= (select pl_team_stage.team_sk from
pl_team_stage where pl_team_stage.TEAM_ID=pl_fact_stage.TEAM_BID);

update pl_fact_stage
set stadium_sk= (select pl_stadium_stage.stadium_sk from
pl_stadium_stage where pl_stadium_stage.TEAM_ID=pl_fact_stage.TEAM_AID);
COMMIT;
END;
```

After that I created the dimensional tables and moved the data from the stage tables into their corresponding dimensional table.



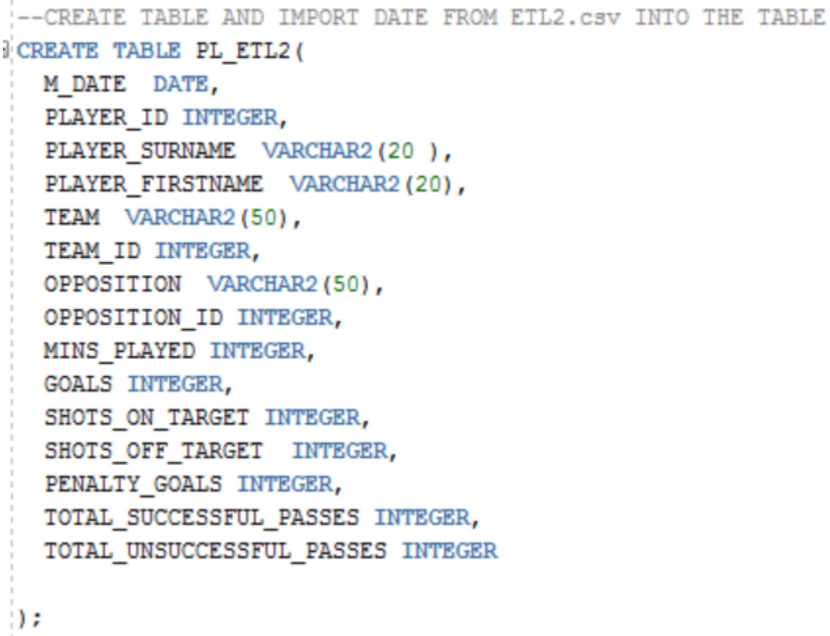
```
create table PL_DIM_Fact
(
  time_sk           INTEGER NOT NULL,
  player_sk         INTEGER NOT NULL,
  team_sk           INTEGER NOT NULL,
  opponent_sk       INTEGER NOT NULL,
  stadium_sk        INTEGER NOT NULL,
  min_played        INTEGER NULL,
  goals             INTEGER NULL,
  shot_on           INTEGER NULL,
  shot_off          INTEGER NULL,
  penalty           INTEGER NULL,
  pass_complete     INTEGER NULL,
  pass_incomplete   INTEGER NULL,
  PRIMARY KEY (time_sk,player_sk,team_sk,opponent_sk,stadium_sk),
  FOREIGN KEY (time_sk) REFERENCES PL_DIM_TIME(time_sk),
  FOREIGN KEY (player_sk) REFERENCES PL_DIM_PLAYER(player_sk),
  FOREIGN KEY (team_sk) REFERENCES PL_DIM_TEAM(team_sk),
  FOREIGN KEY (opponent_sk) REFERENCES PL_DIM_TEAM(team_sk),
  FOREIGN KEY (stadium_sk) REFERENCES PL_DIM_STADIUM(stadium_sk)
);
```



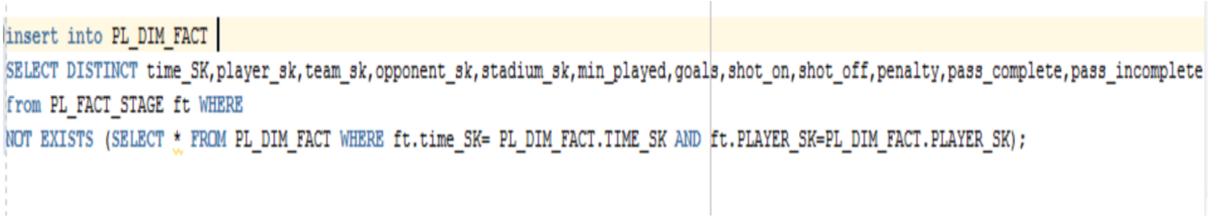
```
insert into PL_DIM_FACT select time_SK,player_sk,team_sk,opponent_sk,stadium_sk,min_played,goals,shot_on,shot_off,penalty,pass_complete,pass_incomplete
from PL_FACT_STAGE ft
where NOT EXISTS (SELECT * FROM PL_DIM_FACT WHERE ft.time_SK= PL_DIM_FACT.TIME_SK AND ft.PLAYER_SK=PL_DIM_FACT.PLAYER_SK);
```

PART D

In part D I created a denormalized table like in part A and imported data from etl2.sql into the table. From there I moved data from the denormalized tables into the stage tables. I recalled the procedure for the fact stage table and then moved all the new data over to the dim tables. Using NOT EXISTS to make sure that there would be no dupes in the dim table from the new set of players.



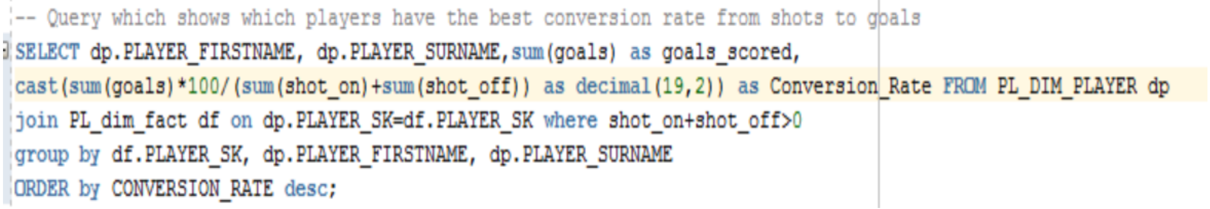
```
--CREATE TABLE AND IMPORT DATE FROM ETL2.csv INTO THE TABLE
CREATE TABLE PL_ETL2(
  M_DATE  DATE,
  PLAYER_ID INTEGER,
  PLAYER_SURNAME  VARCHAR2(20 ),
  PLAYER_FIRSTNAME  VARCHAR2(20),
  TEAM  VARCHAR2(50),
  TEAM_ID INTEGER,
  OPPOSITION  VARCHAR2(50),
  OPPOSITION_ID INTEGER,
  MINS_PLAYED INTEGER,
  GOALS INTEGER,
  SHOTS_ON_TARGET INTEGER,
  SHOTS_OFF_TARGET  INTEGER,
  PENALTY_GOALS INTEGER,
  TOTAL_SUCCESSFUL_PASSES INTEGER,
  TOTAL_UNSUCCESSFUL_PASSES INTEGER

);
```



```
insert into PL_DIM_FACT
SELECT DISTINCT time_SK,player_sk,team_sk,opponent_sk,stadium_sk,min_played,goals,shot_on,shot_off,penalty,pass_complete,pass_incomplete
from PL_FACT_STAGE ft WHERE
NOT EXISTS (SELECT * FROM PL_DIM_FACT WHERE ft.time_SK= PL_DIM_FACT.TIME_SK AND ft.PLAYER_SK=PL_DIM_FACT.PLAYER_SK);
```

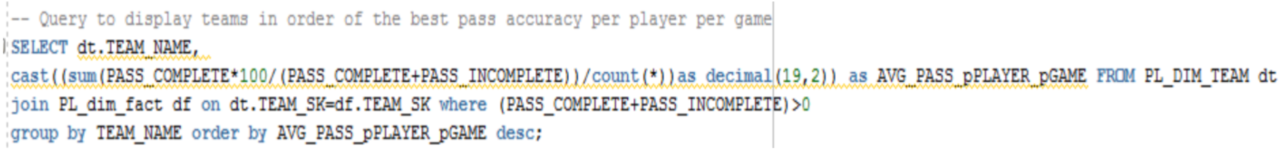
PART E

For the two queries I had to create which can be executed over the dimensional model to create reports about teams and players I chose to create one to show which players have the best conversion rate i.e. which player has the best goal to shot ratio. E.g. a player who has scored 3 goals and taken a total of 6 shots has a conversion rate of 50%. The query takes the sum of each player's goals from every game and divides it by the sum of shots on and off target in every game the player has played. This is multiplied by 100 to get a percentage. There is a where clause which specifies that if a player hasn't had any shots on or off target that it is not included as no player has scored with no shots.



```
-- Query which shows which players have the best conversion rate from shots to goals
SELECT dp.PLAYER_FIRSTNAME, dp.PLAYER_SURNAME,sum(goals) as goals_scored,
cast(sum(goals)*100/(sum(shot_on)+sum(shot_off)) as decimal(19,2)) as Conversion_Rate FROM PL_DIM_PLAYER dp
join PL_dim_fact df on dp.PLAYER_SK=df.PLAYER_SK where shot_on+shot_off>0
group by df.PLAYER_SK, dp.PLAYER_FIRSTNAME, dp.PLAYER_SURNAME
ORDER by CONVERSION_RATE desc;
```

The second query I made is to create a report about teams. This query returns the average pass accuracy per player per game. The query takes the pass accuracy of each player in every match they played and adds them all together for each team. This total is divided by the amount of players matches in the sum (count(*)). This will return the average pass accuracy for each team, and it is sorted in descending order so the team with the best average pass accuracy per player per game is at the top and that team is Manchester United.



```
-- Query to display teams in order of the best pass accuracy per player per game
SELECT dt.TEAM_NAME,
cast((sum(PASS_COMPLETE*100/(PASS_COMPLETE+PASS_INCOMPLETE))/count(*))as decimal(19,2)) as AVG_PASS_pPLAYER_pGAME FROM PL_DIM_TEAM dt
join PL_dim_fact df on dt.TEAM_SK=df.TEAM_SK where (PASS_COMPLETE+PASS_INCOMPLETE)>0
group by TEAM_NAME order by AVG_PASS_pPLAYER_pGAME desc;
```

Summary

To summarise the assignment we were required to complete five tasks which were;

a) Create a relational model using Oracle or MySQL. Implementing the primary and foreign key constraints.
b) Populate the data by loading the data from the sql script and from the csv into the relational model.
c) Perform an initial ETL process to move data into the data warehouse. Define a sql script that automatically executes the process, creating the required stage tables and all the data matching procedures.
d) Perform a second ETL using the additional data contained in the etl2.csv files. Load the data into some staging table and then perform the second ETL reusing some of the stage tables and procedures used for the first ETL.
e) Provide 2 sample queries that could be executed over the dimensional model to create reports about teams and players.

I feel I have completed the required tasks with each part working as it is expected to.