XJTLU Entrepreneur College (Taicang) Cover Sheet

Module code and Title	DTS106TC: Introduction to Database	
School Title	School of AI and Advanced Computing	
Assignment Title	Assessment Task 001 (CW): Individual Coursework	
Submission Deadline	**17 May 2023 at 5:00 PM**	
Final Word Count	NA	
If you agree to let the university use your work anonymously for teaching and learning purposes, please type **"yes"** here.		**Yes**

I certify that I have read and understood the University's Policy for dealing with Plagiarism, Collusion and the Fabrication of Data (available on Learning Mall Online). With reference to this policy I certify that:

- My work does not contain any instances of plagiarism and/or collusion.
  My work does not contain any fabricated data.

By uploading my assignment onto Learning Mall Online, I formally declare that all of the above information is true to the best of my knowledge and belief.

Scoring – For Tutor Use	
Student ID	**2142116**

Stage of Marking	**Marker Code**	**Learning Outcomes Achieved （F/P/M/D） (please modify as appropriate)**				**Final Score**
		A	**B**	**C**	**D**	
1st Marker – red pen						
Moderation – green pen	**IM Initials**	The original mark has been accepted by the moderator (please circle as appropriate):				Y / N
		Data entry and score calculation have been checked by another tutor (please circle):				Y
2nd Marker if needed – green pen						

For Academic Office Use			**Possible Academic Infringement (please tick as appropriate)**	
Date Received	**Days late**	**Late Penalty**	☐ **Category A**	Total Academic Infringement Penalty (A,B, C, D, E, Please modify where necessary) _____
			☐ **Category B**	
			☐ **Category C**	
			☐ **Category D**	
			☐ **Category E**	

Students

(Please modify where necessary)

The assignment must be typed in an MS Word document and submitted as a pdf via Learning Mall Online to the correct dropbox. Only electronic submission is accepted and no hard copy submission.

All students must download their file and check that it is viewable after submission. Documents may become corrupted during the uploading process (e.g. due to slow internet connections). However, students themselves are responsible for submitting a functional and correct file for assessments.

Q1: Requirement Description

The League of Legends mid-season game is underway. In order to let the audience better understand each team, our client wants to create a database to collect relevant information. The database will include the following components:

1. Team Table: This table will contain information about the participating teams, including team name, team logo, and region.

2. Player Table: This table will store details about the players, such as their names, nationalities, positions, MVP count, average KDA, preferred champions, and the team they belong to.

3. Coach Table: This table will store details about the coaches, including their names, nationalities, and the team they are associated with.

4. Match Table: This table will record match-specific information, including match ID, match time, participating teams, and match result.

5. Performance Table: It will include attributes such as performance ID (primary key), ranking, points, win rate, and average game duration. The performance ID will be added as a foreign key in the team table.

Business Rules:

1. Each team can have multiple players, and each player belongs to only one team.

2. Each team can have one coach, and a coach can be associated with only one team.

3. Each hero can have multiple appearances in matches, and their performance statistics will be recorded.

4. Each match will have two participating teams and a result.

Assumptions:

1. The database will be designed to handle data related to a specific League of Legends tournament or season.

2. The data in the database will be regularly updated to reflect the latest tournament results.

3. The performance table will capture the overall performance of each team in the tournament.

Problems:

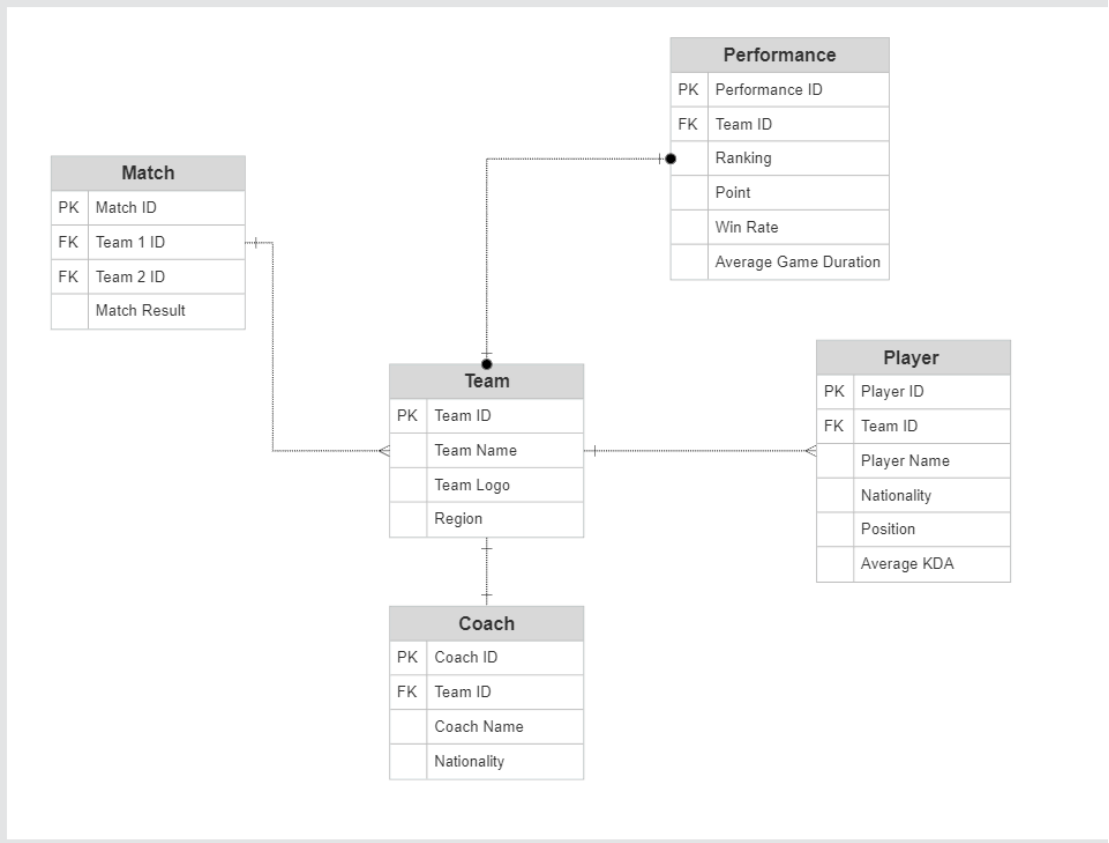
1. How to ensure data integrity and consistency when updating match results and performance statistics.

2. How to handle changes in team compositions, such as player transfers or coaching staff changes.

3. How to efficiently query and analyze team performances based on the provided data.

By addressing these business requirements, establishing relationships between the tables, and considering the business rules and assumptions, the database will provide a solid foundation for managing and analyzing tournament-related data in the League of Legends Midseason.

Q2: Conceptual Model



The selected design is suitable for several reasons:

1. Normalization: The design follows normalization principles, ensuring that the database is structured efficiently, with minimal data redundancy and consistent relationships between entities. This helps to maintain data integrity and optimize query performance.

2. Data Consistency: By properly defining primary keys and foreign keys, the design enforces data consistency and referential integrity. The relationships between entities ensure that data dependencies are accurately represented and maintained throughout the database.

3. Flexibility: The design allows for scalability and adaptability. It can accommodate multiple teams, players, coaches, matches, and their respective performance records. It provides the necessary structure to store and retrieve data effectively, enabling future expansion and modifications.

4. Query Optimization: The design supports efficient querying and analysis of data. By appropriately linking entities through foreign keys, it enables straightforward joins and retrieval of related information. This helps in generating reports, analyzing team performance, and making data-driven decisions.

Regarding alternative designs, some possibilities could include:

1. Denormalization: Instead of normalizing the data into separate tables, one could combine multiple entities into a single table to simplify the structure. However, this could lead to data redundancy, update anomalies, and decreased performance, especially when dealing with complex relationships and large datasets.

2. Embedded Attributes: Instead of having separate tables for players and coaches, their attributes could be directly included as columns in the team table. However, this would limit the flexibility to handle multiple players or coaches associated with a team. Coach and player turnovers are a common occurrence, and putting them all in one update is not good for the database.

3. Performance Table per Team: Instead of having a single performance table, each team could have its own performance table. However, this approach would result in a larger number of tables, making data management and analysis more complex. It would also hinder the ability to compare and analyze the performance of multiple teams simultaneously.

These alternative designs were not chosen for the final design because they would compromise the principles of normalization, data consistency, flexibility, and query optimization. The selected design strikes a balance between these factors and provides a robust foundation for managing and analyzing the tournament-related data effectively.

Q3: Logical Model

Yes, I remember the ERD model we discussed earlier. Here's a description of the design for each table in the ERD, along with the candidate keys, functional dependencies, and assumptions:

1. Team Table:

- Design: This table represents the participating teams in the MSI.

- Attributes: Team ID (Primary Key), Team Name, Team Logo, Region.

- Candidate Key: Team ID.

- Functional Dependencies: None.

- Assumptions: Each team has a unique Team ID. The Team ID will be used as the primary key to identify each team.

Column Name	Team ID	Team Name	Team Logo	Region
Key Type	PK			
Not Null = NN Unique = U	NN, U	NN, U	NN	NN
Data Type	NUMBER	VARCHAR	BLOB	VARCHAR
Length	30	15		15

2. Player Table:

- Design: This table stores information about the players participating in the tournament.

- Attributes: Player ID (Primary Key), Player Name, Nationality, Position, MVP Count, Average KDA, Preferred Champions, Team ID (Foreign Key).

- Candidate Key: Player ID.

- Functional Dependencies: Player ID → Player Name, Nationality, Position, MVP Count, Average KDA, Preferred Champions, Team ID.

- Assumptions: Each player has a unique Player ID. The Player ID will be used as the primary key. Each player belongs to only one team based on the Team ID foreign key and a team have multiple players.

Column Name	Player ID	Team ID	Player Name	Nationality	Position	Average KDA
Key Type	PK	FK				
Not Null = NN Unique = U	NN, U	NN	NN	NN	NN	NN
Data Type	NUMBER	NUMBER	VARCHAR	VARCHAR	VARCHAR	FLOAT
Length	30	30	15	15	15	

3. Coach Table:

- Design: This table stores details about the coaches associated with the teams.

- Attributes: Coach ID (Primary Key), Coach Name, Nationality, Team ID (Foreign Key).

- Candidate Key: Coach ID.

- Functional Dependencies: Coach ID → Coach Name, Nationality, Team ID.

- Assumptions: Each coach has a unique Coach ID. The Coach ID will be used as the primary key. Each coach is associated with only one team based on the Team ID foreign key and there is only one coach per team

Column Name	Coach ID	Team ID	Coach Name	Nationality
Key Type	PK	FK		
Not Null = NN Unique = U	NN, U	NN, U	NN, U	NN
Data Type	NUMBER	NUMBER	VARCHAR	VARCHAR
Length	30	30	15	15

4. Match Table:

- Design: This table records the details of the matches in the tournament.

- Attributes: Match ID (Primary Key), Match Time, Team 1 ID (Foreign Key), Team 2 ID (Foreign Key), Match Result.

- Candidate Key: Match ID.

- Functional Dependencies: Match ID → Match Time, Team 1 ID, Team 2 ID, Match Result.

- Assumptions: Each match has a unique Match ID. The Match ID will be used as the primary key. Each match involves two teams, identified by the Team 1 ID and Team 2 ID foreign keys.

Column Name	Match ID	Match Time	Team1 ID	Team2 ID	Match Result
Key Type	PK		FK	FK	
Not Null = NN Unique = U	NN, U	NN	NN	NN	NN
Data Type	NUMBER	DATE	NUMBER	NUMBER	VARCHAR
Length	30		30	30	15

5. Performance Table:

- Design: This table represents the overall performance of each team in the tournament.

- Attributes: Performance ID (Primary Key), Ranking, Points, Win Rate, Average Game Duration, Team ID (Foreign Key).

- Candidate Key: Performance ID.

- Functional Dependencies: Performance ID → Ranking, Points, Win Rate, Average Game Duration, Team ID.

- Assumptions: Each team has a unique performance record. The Performance ID will be used as the primary key. The Team ID foreign key establishes the relationship between the performance and the corresponding team.

Column Name	Performance ID	Team ID	Ranking	Point	Win Rate	Average Game Duration
Key Type	PK	FK				
Not Null = NN Unique = U	NN, U	NN, U				
Data Type	NUMBER	NUMBER	NUMBER	NUMBER	FLOAT	FLOAT
Length	30	30	30	30		

Regarding other possible designs, one alternative could be to have separate tables for each region in the Team Table, such as North America, Europe, Korea, etc. However, this design would result in a larger number of tables and could complicate querying and reporting when analyzing teams collectively. The

chosen design of having a single Team Table with a Region attribute allows for better data organization and ease of analysis across different regions.

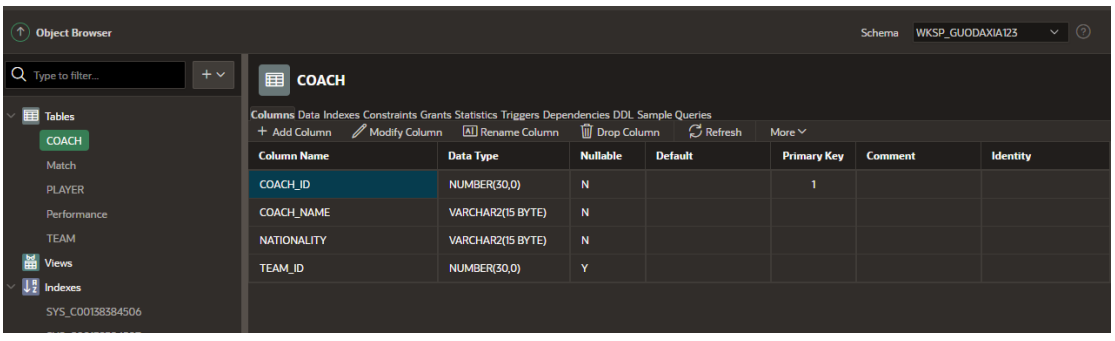
Another possible design could involve denormalizing the Player Table by directly including the Team Name attribute instead of using the Team ID foreign key. However, this would introduce data redundancy and make updates and maintenance more complex. The chosen design of using the Team ID foreign key ensures data consistency and easier management of team-player relationships.

Overall, the selected design balances the trade-offs between data organization, performance, and maintainability, while allowing for flexibility and scalability in representing the League of Legends tournament data.

Q4: Physical Model.

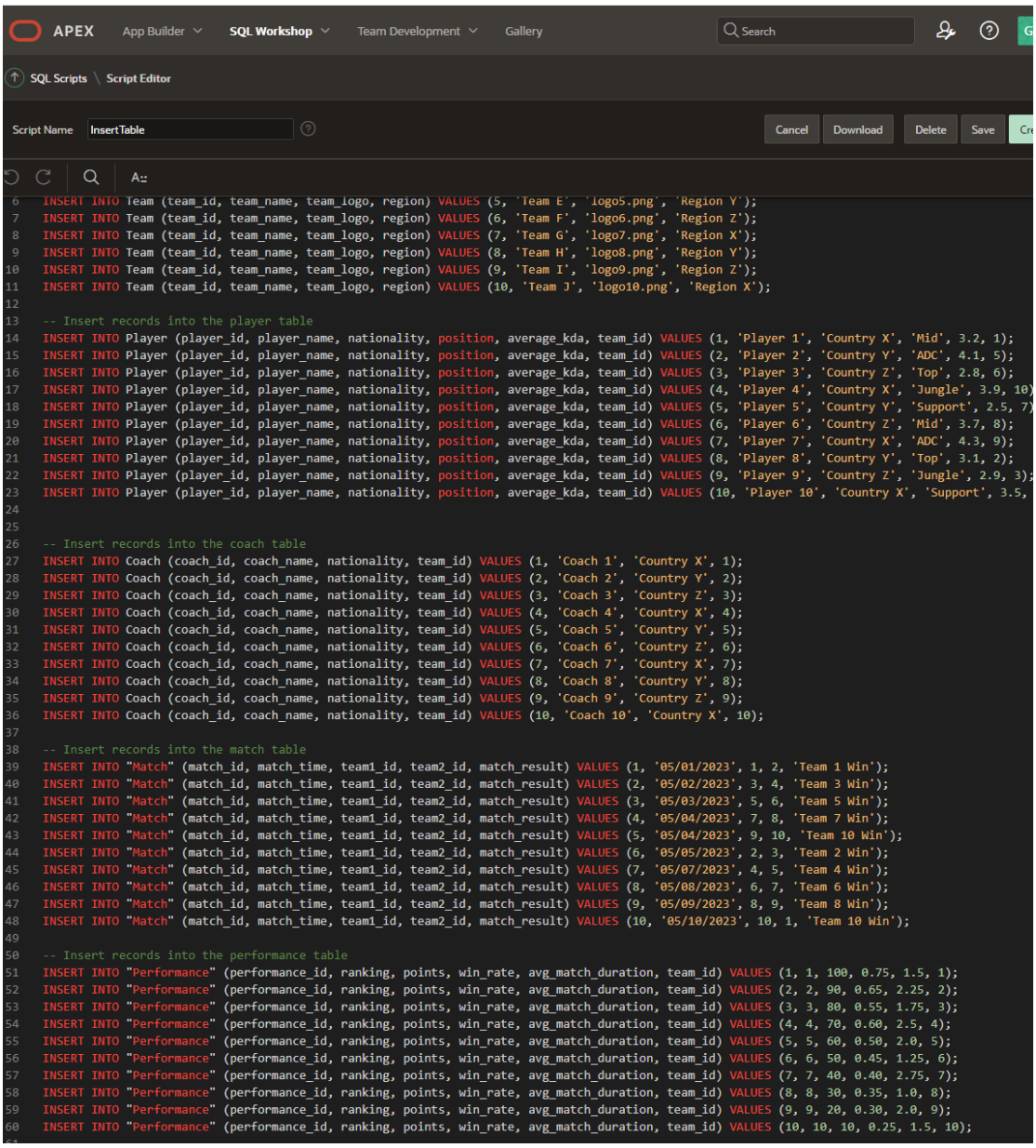
4a) Create a Table for each entity using Oracle Apex, as shown in the following figure. The constraints correspond to the table in q3. The team logo should be in BLOB format, using Varchar format for the convenience of inserting values.

```sql
DROP TABLE "Performance";
DROP TABLE "Match";
DROP TABLE Coach;
DROP TABLE Player;
DROP TABLE Team;

CREATE TABLE Team (
  team_id NUMBER(30) PRIMARY KEY,
  team_name VARCHAR(15) NOT NULL UNIQUE,
  team_logo VARCHAR(15) NOT NULL,
  region VARCHAR(15) NOT NULL
);

CREATE TABLE Player (
  player_id NUMBER(30) PRIMARY KEY,
  player_name VARCHAR(15) NOT NULL,
  nationality VARCHAR(15) NOT NULL,
  position VARCHAR(15) NOT NULL,
  average_kda FLOAT NOT NULL,
  team_id NUMBER(30),
  FOREIGN KEY (team_id) REFERENCES Team(team_id)
);


CREATE TABLE Coach (
  coach_id NUMBER(30) PRIMARY KEY,
  coach_name VARCHAR(15) NOT NULL,
  nationality VARCHAR(15) NOT NULL,
  team_id NUMBER(30) UNIQUE,
  FOREIGN KEY (team_id) REFERENCES Team(team_id)
);


CREATE TABLE "Match" (
  match_id NUMBER(30) PRIMARY KEY,
  match_time DATE NOT NULL,
  team1_id NUMBER(30),
  team2_id NUMBER(30),
  match_result VARCHAR(15) NOT NULL,
  FOREIGN KEY (team1_id) REFERENCES Team(team_id),
  FOREIGN KEY (team2_id) REFERENCES Team(team_id)
);


CREATE TABLE "Performance" (
  performance_id NUMBER(30) PRIMARY KEY,
  ranking NUMBER(30),
  points NUMBER(30),
  win_rate FLOAT,
  average_game_duration FLOAT,
  team_id NUMBER(30) UNIQUE,
  FOREIGN KEY (team_id) REFERENCES Team(team_id)
  );

```

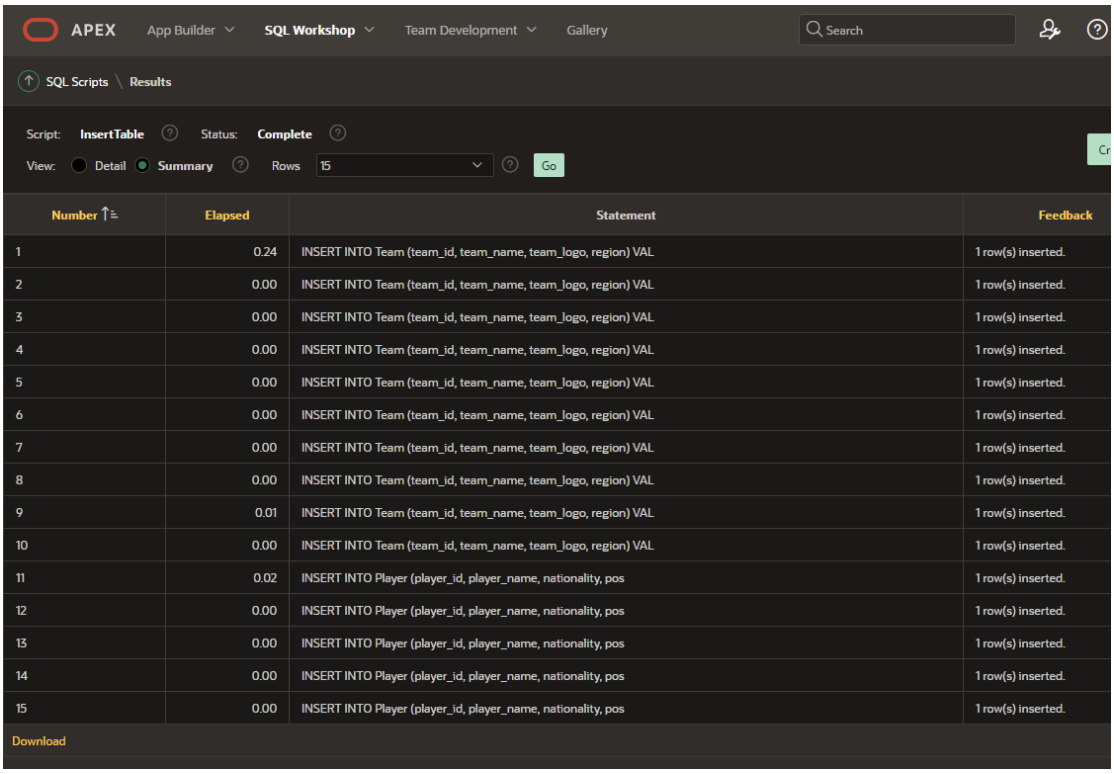


4b)

Use the INSERT statement to insert some example data. All the foreign keys are derived from the primary key of the Team table, so we need to populate the Team table first.



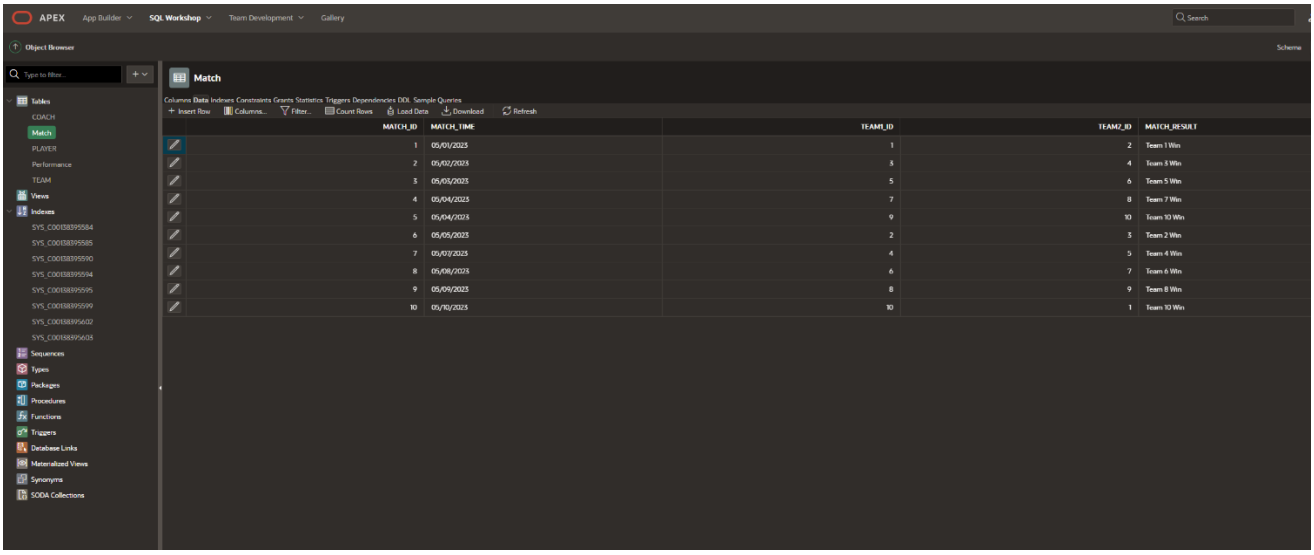

```sql
INSERT INTO Team (team_id, team_name, team_logo, region) VALUES (5, 'Team E', 'logo5.png', 'Region Y');
INSERT INTO Team (team_id, team_name, team_logo, region) VALUES (6, 'Team F', 'logo6.png', 'Region Z');
INSERT INTO Team (team_id, team_name, team_logo, region) VALUES (7, 'Team G', 'logo7.png', 'Region X');
INSERT INTO Team (team_id, team_name, team_logo, region) VALUES (8, 'Team H', 'logo8.png', 'Region Y');
INSERT INTO Team (team_id, team_name, team_logo, region) VALUES (9, 'Team I', 'logo9.png', 'Region Z');
INSERT INTO Team (team_id, team_name, team_logo, region) VALUES (10, 'Team J', 'logo10.png', 'Region X');

-- Insert records into the player table
INSERT INTO Player (player_id, player_name, nationality, position, average_kda, team_id) VALUES (1, 'Player 1', 'Country X', 'Mid', 3.2, 1);
INSERT INTO Player (player_id, player_name, nationality, position, average_kda, team_id) VALUES (2, 'Player 2', 'Country Y', 'ADC', 4.1, 5);
INSERT INTO Player (player_id, player_name, nationality, position, average_kda, team_id) VALUES (3, 'Player 3', 'Country Z', 'Top', 2.8, 6);
INSERT INTO Player (player_id, player_name, nationality, position, average_kda, team_id) VALUES (4, 'Player 4', 'Country X', 'Jungle', 3.9, 10)
INSERT INTO Player (player_id, player_name, nationality, position, average_kda, team_id) VALUES (5, 'Player 5', 'Country Y', 'Support', 2.5, 7)
INSERT INTO Player (player_id, player_name, nationality, position, average_kda, team_id) VALUES (6, 'Player 6', 'Country Z', 'Mid', 3.7, 8);
INSERT INTO Player (player_id, player_name, nationality, position, average_kda, team_id) VALUES (7, 'Player 7', 'Country X', 'ADC', 4.3, 9);
INSERT INTO Player (player_id, player_name, nationality, position, average_kda, team_id) VALUES (8, 'Player 8', 'Country Y', 'Top', 3.1, 2);
INSERT INTO Player (player_id, player_name, nationality, position, average_kda, team_id) VALUES (9, 'Player 9', 'Country Z', 'Jungle', 2.9, 3);
INSERT INTO Player (player_id, player_name, nationality, position, average_kda, team_id) VALUES (10, 'Player 10', 'Country X', 'Support', 3.5,


-- Insert records into the coach table
INSERT INTO Coach (coach_id, coach_name, nationality, team_id) VALUES (1, 'Coach 1', 'Country X', 1);
INSERT INTO Coach (coach_id, coach_name, nationality, team_id) VALUES (2, 'Coach 2', 'Country Y', 2);
INSERT INTO Coach (coach_id, coach_name, nationality, team_id) VALUES (3, 'Coach 3', 'Country Z', 3);
INSERT INTO Coach (coach_id, coach_name, nationality, team_id) VALUES (4, 'Coach 4', 'Country X', 4);
INSERT INTO Coach (coach_id, coach_name, nationality, team_id) VALUES (5, 'Coach 5', 'Country Y', 5);
INSERT INTO Coach (coach_id, coach_name, nationality, team_id) VALUES (6, 'Coach 6', 'Country Z', 6);
INSERT INTO Coach (coach_id, coach_name, nationality, team_id) VALUES (7, 'Coach 7', 'Country X', 7);
INSERT INTO Coach (coach_id, coach_name, nationality, team_id) VALUES (8, 'Coach 8', 'Country Y', 8);
INSERT INTO Coach (coach_id, coach_name, nationality, team_id) VALUES (9, 'Coach 9', 'Country Z', 9);
INSERT INTO Coach (coach_id, coach_name, nationality, team_id) VALUES (10, 'Coach 10', 'Country X', 10);

-- Insert records into the match table
INSERT INTO "Match" (match_id, match_time, team1_id, team2_id, match_result) VALUES (1, '05/01/2023', 1, 2, 'Team 1 Win');
INSERT INTO "Match" (match_id, match_time, team1_id, team2_id, match_result) VALUES (2, '05/02/2023', 3, 4, 'Team 3 Win');
INSERT INTO "Match" (match_id, match_time, team1_id, team2_id, match_result) VALUES (3, '05/03/2023', 5, 6, 'Team 5 Win');
INSERT INTO "Match" (match_id, match_time, team1_id, team2_id, match_result) VALUES (4, '05/04/2023', 7, 8, 'Team 7 Win');
INSERT INTO "Match" (match_id, match_time, team1_id, team2_id, match_result) VALUES (5, '05/04/2023', 9, 10, 'Team 10 Win');
INSERT INTO "Match" (match_id, match_time, team1_id, team2_id, match_result) VALUES (6, '05/05/2023', 2, 3, 'Team 2 Win');
INSERT INTO "Match" (match_id, match_time, team1_id, team2_id, match_result) VALUES (7, '05/07/2023', 4, 5, 'Team 4 Win');
INSERT INTO "Match" (match_id, match_time, team1_id, team2_id, match_result) VALUES (8, '05/08/2023', 6, 7, 'Team 6 Win');
INSERT INTO "Match" (match_id, match_time, team1_id, team2_id, match_result) VALUES (9, '05/09/2023', 8, 9, 'Team 8 Win');
INSERT INTO "Match" (match_id, match_time, team1_id, team2_id, match_result) VALUES (10, '05/10/2023', 10, 1, 'Team 10 Win');

-- Insert records into the performance table
INSERT INTO "Performance" (performance_id, ranking, points, win_rate, avg_match_duration, team_id) VALUES (1, 1, 100, 0.75, 1.5, 1);
INSERT INTO "Performance" (performance_id, ranking, points, win_rate, avg_match_duration, team_id) VALUES (2, 2, 90, 0.65, 2.25, 2);
INSERT INTO "Performance" (performance_id, ranking, points, win_rate, avg_match_duration, team_id) VALUES (3, 3, 80, 0.55, 1.75, 3);
INSERT INTO "Performance" (performance_id, ranking, points, win_rate, avg_match_duration, team_id) VALUES (4, 4, 70, 0.60, 2.5, 4);
INSERT INTO "Performance" (performance_id, ranking, points, win_rate, avg_match_duration, team_id) VALUES (5, 5, 60, 0.50, 2.0, 5);
INSERT INTO "Performance" (performance_id, ranking, points, win_rate, avg_match_duration, team_id) VALUES (6, 6, 50, 0.45, 1.25, 6);
INSERT INTO "Performance" (performance_id, ranking, points, win_rate, avg_match_duration, team_id) VALUES (7, 7, 40, 0.40, 2.75, 7);
INSERT INTO "Performance" (performance_id, ranking, points, win_rate, avg_match_duration, team_id) VALUES (8, 8, 30, 0.35, 1.0, 8);
INSERT INTO "Performance" (performance_id, ranking, points, win_rate, avg_match_duration, team_id) VALUES (9, 9, 20, 0.30, 2.0, 9);
INSERT INTO "Performance" (performance_id, ranking, points, win_rate, avg_match_duration, team_id) VALUES (10, 10, 10, 0.25, 1.5, 10);
```



Number	Elapsed	Statement	Feedback
1	0.24	INSERT INTO Team (team_id, team_name, team_logo, region) VAL	1 row(s) inserted.
2	0.00	INSERT INTO Team (team_id, team_name, team_logo, region) VAL	1 row(s) inserted.
3	0.00	INSERT INTO Team (team_id, team_name, team_logo, region) VAL	1 row(s) inserted.
4	0.00	INSERT INTO Team (team_id, team_name, team_logo, region) VAL	1 row(s) inserted.
5	0.00	INSERT INTO Team (team_id, team_name, team_logo, region) VAL	1 row(s) inserted.
6	0.00	INSERT INTO Team (team_id, team_name, team_logo, region) VAL	1 row(s) inserted.
7	0.00	INSERT INTO Team (team_id, team_name, team_logo, region) VAL	1 row(s) inserted.
8	0.00	INSERT INTO Team (team_id, team_name, team_logo, region) VAL	1 row(s) inserted.
9	0.01	INSERT INTO Team (team_id, team_name, team_logo, region) VAL	1 row(s) inserted.
10	0.00	INSERT INTO Team (team_id, team_name, team_logo, region) VAL	1 row(s) inserted.
11	0.02	INSERT INTO Player (player_id, player_name, nationality, pos	1 row(s) inserted.
12	0.00	INSERT INTO Player (player_id, player_name, nationality, pos	1 row(s) inserted.
13	0.00	INSERT INTO Player (player_id, player_name, nationality, pos	1 row(s) inserted.
14	0.00	INSERT INTO Player (player_id, player_name, nationality, pos	1 row(s) inserted.
15	0.00	INSERT INTO Player (player_id, player_name, nationality, pos	1 row(s) inserted.

Download

Now let's take a look at the results. Taking 'Match' as an example, we can see that the data has been correctly inserted into the table.



Q5: SQL Queries

1. Query to retrieve the team names, total number of players, and number of coaches for each team

(For convenience, I only set up one player per team):

```sql
SELECT t.team_name, COUNT(p.player_id) AS total_players, COUNT(c.coach_id) AS total_coaches
FROM Team t
LEFT JOIN Player p ON t.team_id = p.team_id
LEFT JOIN Coach c ON t.team_id = c.team_id
GROUP BY t.team_name;
```

Results | Explain | Describe | Saved SQL | History

TEAM_NAME	TOTAL_PLAYERS	TOTAL_COACHES
Team D	1	1
Team I	1	1
Team J	1	1
Team C	1	1
Team A	1	1
Team B	1	1
Team H	1	1
Team E	1	1
Team F	1	1
Team G	1	1

2. Query to retrieve the team names and information of the player with the highest average KDA for each team:

```sql
SELECT t.team_name, p.player_name, p.average_kda
FROM Team t
INNER JOIN Player p ON t.team_id = p.team_id
WHERE p.average_kda = (
  SELECT MAX(average_kda)
  FROM Player
  WHERE team_id = t.team_id
);
```

Results | Explain | Describe | Saved SQL | History

TEAM_NAME	PLAYER_NAME	AVERAGE_KDA
Team A	Player 1	3.2
Team B	Player 8	3.1
Team C	Player 9	2.9
Team D	Player 10	3.5
Team E	Player 2	4.1
Team F	Player 3	2.8
Team G	Player 5	2.5
Team H	Player 6	3.7
Team I	Player 7	4.3
Team J	Player 4	3.9

3. Query to calculate the average KDA of all players in a specific team:

```sql
SELECT AVG(average_kda) AS average_kda
FROM Player
WHERE team_id = 1;
```

Results | Explain | Describe | Saved SQL | History

AVERAGE_KDA
3.2

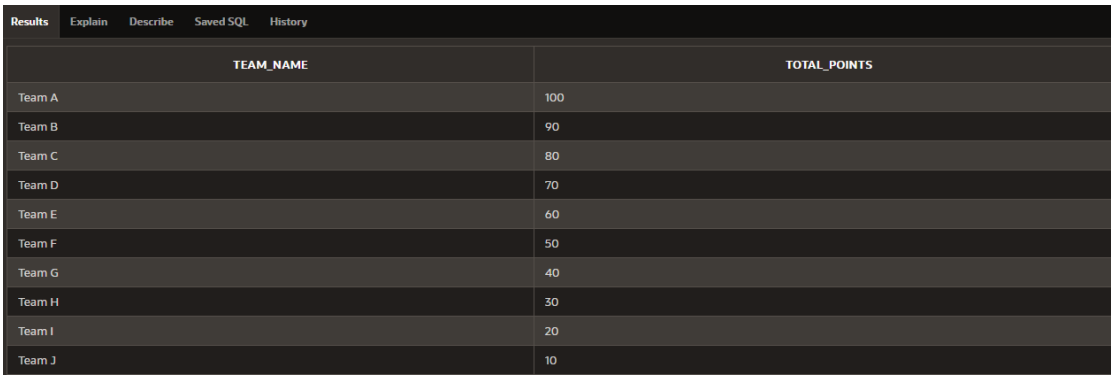
4. Query to retrieve the match date, team names of participating teams, and match result for each match (There seems to be some problems here, the result should be similar to "Team A win", and it is because the data inserted into the table is given by me at will. But my query is fine):

```sql
SELECT m.match_time, t1.team_name AS team1_name, t2.team_name AS team2_name, m.match_result
FROM "Match" m
INNER JOIN Team t1 ON m.team1_id = t1.team_id
INNER JOIN Team t2 ON m.team2_id = t2.team_id;
```

MATCH_TIME	TEAM1_NAME	TEAM2_NAME	MATCH_RESULT
05/10/2023	Team J	Team A	Team 10 Win
05/01/2023	Team A	Team B	Team 1 Win
05/05/2023	Team B	Team C	Team 2 Win
05/02/2023	Team C	Team D	Team 3 Win
05/07/2023	Team D	Team E	Team 4 Win
05/03/2023	Team E	Team F	Team 5 Win
05/08/2023	Team F	Team G	Team 6 Win
05/04/2023	Team G	Team H	Team 7 Win
05/09/2023	Team H	Team I	Team 8 Win
05/04/2023	Team I	Team J	Team 10 Win

5. Query to retrieve the team names and total points for each team, and rank them by points (descending):

```sql
SELECT t.team_name, COALESCE(SUM(p.points), 0) AS total_points
FROM Team t
LEFT JOIN "Performance" p ON t.team_id = p.team_id
GROUP BY t.team_name
ORDER BY total_points DESC;
```



Results | Explain | Describe | Saved SQL | History

TEAM_NAME	TOTAL_POINTS
Team A	100
Team B	90
Team C	80
Team D	70
Team E	60
Team F	50
Team G	40
Team H	30
Team I	20
Team J	10

Q6: Relational Algebra Queries

1. Query: Retrieve the team names and their corresponding coach names.
Relational Algebra:

π team_name, coach_name (Team ⋈ Coach)

The result may be:

Team	Coach
Team A	Coach A
Team B	Coach B
Team C	Coach C

2. Query: Calculate the average KDA for each position across all teams.
Relational Algebra:

π position, AVG(average_kda) (Player/position)

The result may be:

Position	Avg_kda
Mid	3.2
ADC	4.0
Top	2.8
Jungle	3.9
Support	2.5

3. Query：Calculate the average winning percentage of each division。
Relational Algebra:

π region, AVG(win_rate) (Team ⋈ Performance)

The result may be:

Region	AVG(win_rate)
Region X	0.65
Region Y	0.72

Region Z	0.68

4. Query: Retrieve the team names and their corresponding coach names for teams that have higher average match duration than the overall average match duration.

Relational Algebra:

π team_name, coach_name ((Team ⋈ Performance) ⋈ (π AVG(avg_match_duration) (Performance)) | avg_match_duration > AVG(avg_match_duration))

The result may be:

Team	Coach
Team A	Coach1
Team C	Coach3

5. Query: Find the player with the highest average KDA in each division and the name of their team.

Relational Algebra:

π region, player_name, team_name (Player ⋈ (π region, MAX(average_kda) (Player ⋈ Team)) ⋈ Team)

The result may be:

Region	Player	Team
Region X	Player1	Team A
Region Y	Player2	Team B
Region Z	Player3	Team C