**View Statements**

CREATE VIEW view\_1 AS SELECT pla\_first, pla\_last,pla\_year, pla\_GPA

FROM players

SELECT \*

FROM view\_1;

This view would be used by coaches to determine any trends with their players depending on their year. It can also be used to determine which class year has the highest average GPA.

CREATE VIEW view\_2 AS SELECT coa\_first, coa\_last, coa\_email, coa\_phone\_number

FROM coaches;

SELECT \*

FROM view\_2;

This view would be used by players on the team and players being recruited to identify their coaches' contact information. Recruits would be able to easily locate all of the coaches' information from each team.

**Functions**

--This function will count the players at a specific school

CREATE OR REPLACE FUNCTION count\_players (school VARCHAR(20))

RETURNS INTEGER

LANGUAGE plpgsql

AS

$$

DECLARE

message VARCHAR(100) = 'Name entered is not a school in this database.';

BEGIN

IF school = 'Eastern' THEN

RETURN (SELECT COUNT(pla\_ID) AS player\_count

FROM players, universities

WHERE universities.uni\_ID = players.uni\_ID AND uni\_name = 'Eastern'

GROUP BY uni\_name);

END IF;

IF school = 'York' THEN

RETURN (SELECT COUNT(pla\_ID) AS player\_count

FROM players, universities

WHERE universities.uni\_ID = players.uni\_ID AND uni\_name = 'York'

GROUP BY uni\_name);

END IF;

IF school = 'Alvernia' THEN

RETURN (SELECT COUNT(pla\_ID) AS player\_count

FROM players, universities

WHERE universities.uni\_ID = players.uni\_ID AND uni\_name = 'Alvernia'

GROUP BY uni\_name);

END IF;

IF school = 'Albright' THEN

RETURN (SELECT COUNT(pla\_ID) AS player\_count

FROM players, universities

WHERE universities.uni\_ID = players.uni\_ID AND uni\_name = 'Albright'

GROUP BY uni\_name);

END IF;

IF school = 'Messiah' THEN

RETURN (SELECT COUNT(pla\_ID) AS player\_count

FROM players, universities

WHERE universities.uni\_ID = players.uni\_ID AND uni\_name = 'Messiah'

GROUP BY uni\_name);

END IF;

END;

$$;

SELECT count\_players('Alvernia');

--This function will delete all the seniors within the database.

CREATE OR REPLACE FUNCTION delete\_seniors()

RETURNS VARCHAR(30)

LANGUAGE plpgsql

AS

$$

BEGIN

RETURN (SELECT COUNT(pla\_ID) AS count\_of\_seniors

FROM players, universities

WHERE universities.uni\_ID = players.uni\_ID AND pla\_year = 4);

DELETE FROM players

WHERE pla\_year = 4;

END;

$$;

SELECT delete\_seniors();

**Procedure**

--This procedure will add a year to all of the players.

CREATE OR REPLACE PROCEDURE addYear()

LANGUAGE plpgsql

AS

$$

BEGIN

UPDATE players

SET pla\_year = pla\_year + 1;

END;

$$

**Trigger**

--This trigger adds a year to all players, whenever new players are added.

CREATE OR REPLACE FUNCTION addYears()

RETURNS TRIGGER

LANGUAGE PLPGSQL

AS

$$

DECLARE

year INTEGER;

BEGIN

UPDATE players

SET pla\_year = pla\_year + 1;

RETURN (SELECT pla\_year

FROM players

WHERE pla\_first = 'Christopher' AND pla\_last = 'Fanelli');

END;

$$;

CREATE TRIGGER addYear

AFTER INSERT ON players

FOR EACH ROW

EXECUTE PROCEDURE addYears();

**Update / Delete Statements:**

UPDATE players

SET pla\_GPA = 3.8

WHERE pla\_last = ‘Gilbert’;

DELETE FROM universities

WHERE uni\_name = ‘Alvernia’;

UPDATE sponsor

SET spo\_name = ‘New Balance’

WHERE spo\_name = ‘Nike’;

DELETE FROM players

WHERE pla\_first = ‘Ryan’ AND pla\_last = ‘Kennedy’;

UPDATE players

SET pla\_year = pla\_year + 1

WHERE pla\_first = ‘Kyran’;

DELETE FROM coaches

WHERE coa\_last = ‘Waldman’;

**Queries:**

-- This query will find all of the players that have a GPA over a 3.6. It will list their name, GPA, and university name.

SELECT pla\_first, pla\_last, pla\_GPA, uni\_name

FROM players, universities

WHERE pla\_GPA > 3.6 AND players.uni\_ID = universities.uni\_ID;

--This query will show how many players play at each respective university, using the count command and the group by command.

SELECT uni\_name, COUNT(pla\_ID) AS Amount

FROM universities, players

WHERE pla\_GPA = 4.0 AND players.uni\_ID = universities.uni\_ID

GROUP BY uni\_name;

--This query will find all of the coaches that have coached for over 10 years within the database.

SELECT coa\_first, coa\_last, coa\_year, uni\_name

FROM coaches, universities

WHERE coa\_year > 10 AND coaches.uni\_ID = universities.uni\_ID;

--This query will find all of the nil deals that are available, while also using order by in order to show the amounts in descending values.

SELECT nil\_name, nil\_amount

FROM nil, players

GROUP BY nil\_name, nil\_amount

ORDER BY nil\_amount DESC;

--This query will show the data from the lifts at eastern university that are of lif\_type 'Weight'.

SELECT lif\_ID, lif\_time, lif\_day, lif\_type

FROM (SELECT lif\_ID, lif\_time, lif\_day, lif\_type

FROM lifts, universities

WHERE uni\_name = 'Eastern' AND universities.uni\_ID = lifts.uni\_ID) AS eastern\_lifts

WHERE lif\_type = 'Weight';

--This query will give the head coaches that have coached at the university for mroe then five years, using the intersect command.

(SELECT coa\_ID, coa\_first, coa\_last

FROM coaches

WHERE coa\_position = 'Head Coach')

INTERSECT

(SELECT coa\_ID, coa\_first, coa\_last

FROM coaches

WHERE coa\_year > 5);

--This query will find when each university has its lifts, while also showing how many players go to each lift.

SELECT uni\_name, lif\_day, COUNT(players.pla\_ID) AS player\_count

FROM players, lifts\_players, lifts, universities

WHERE universities.uni\_ID = players.uni\_ID

AND players.pla\_ID = lifts\_players.pla\_ID

AND lifts\_players.lif\_ID = lifts.lif\_ID

GROUP BY uni\_name, lif\_day

ORDER BY uni\_name;

--Lets see what my GPA will be after this class

SELECT pla\_first, pla\_last, pla\_GPA

FROM players

WHERE pla\_first = 'Nathan' AND pla\_last = 'Gilbert';