CINS 370 Team Project Assignment 4

Team # 1

Aakarshan Bhandari, Kris Selvidge, Min Toe

Table Of Contents

Oracle SQL Commands

Comparing height and weight ranges by position

Description:

This query returns the ranges of the height and weight columns, by displaying the maximum and minimum values, as defined by the player's position. It also returns the average values of the two respective columns. The same solution can be derived from the following two commands.

Join:

```
SELECT
    a.position,
    MAX(a.height) as Max_Height,
    MIN(a.height) as Min_Height,
    CAST(AVG(a.height) as decimal(10,2)) as Avg_Height,
    MAX(a.weight) as Max_Weight,
    MIN(a.weight) as Min_Weight,
    CAST(AVG(a.weight) as decimal(10,2)) as Avg_Weight
FROM player a
INNER JOIN player b ON a.player_code = b.player_code AND a.position IS NOT NULL
GROUP BY a.position
ORDER BY a.position ASC;
```

```
SQL> SELECT
  2      a.position,
  3      MAX(a.height) as Max_Height,
  4      MIN(a.height) as Min_Height,
  5      CAST(AVG(a.height) as decimal(10,2)) as Avg_Height,
  6      MAX(a.weight) as Max_Weight,
  7      MIN(a.weight) as Min_Weight,
  8      CAST(AVG(a.weight) as decimal(10,2)) as Avg_Weight
  9  FROM player a
 10  INNER JOIN player b ON a.player_code = b.player_code AND a.position IS NOT NULL
 11  GROUP BY a.position
 12  ORDER BY a.position ASC;

POSIT MAX_HEIGHT MIN_HEIGHT AVG_HEIGHT MAX_WEIGHT MIN_WEIGHT AVG_WEIGHT
----- ---------- ---------- ---------- ---------- ---------- ----------
ATH           76         68      72.08        225        165     190.62
C             79         71      75.04        345        245     290.09
CB            75         61      70.74        220        150     182.15
DB            78         65      71.42        237        146     188.95
DE            81         70      75.15        336        172     250.42
DL            82         68      74.81        380        172     269.95
DS            78         69      72.75        285        168     222.43
DT            79         70       74.5        380        222     289.11
FB            76         65       72.1        308        198     233.76
FL            75         69      72.31        206        159     182.69
FS            75         69      71.81        212        175     196.08
HB            76         69      72.14        250        175     205.21
HOLD          77         75         76        212        187      199.5
ILB           76         69      73.03        264        185     227.95
K             78         62      71.65        260        125     187.77
LB            78         67      73.16        268        160     223.05
LS            80         66      73.26        310        165     223.82
MLB           74         74         74        250        215        235
NG            76         71      73.96        325        245     291.64
NT            77         72      74.31        323        265      293.5
OG            80         71      75.83        395        240     302.51
OL            81         70      76.12        385        220     295.34
OLB           78         69      74.36        265        191     227.53
OT            83         73      77.37        350        240     296.39
P             80         67      73.47        250        150     199.02
PK            78         67      71.72        240        150      189.1
QB            79         69      74.29        257        153     207.18
RB            76         57      70.29        265        135     201.34
ROV           74         70         72        216        188     201.75
S             77         67      72.11        286        159     196.79
SB            76         66      70.12        250        147     192.64
SE            77         70      73.14        208        180     198.86
SN            76         66      73.32        241        185        218
SS            77         70      72.18        218        174     197.42
TB            75         66      70.64        235        150     201.11
TE            81         71      75.97        307        195     241.08
WR            80         61       72.4        263        138        190

37 rows selected.

SQL>
```

Nested Query:

```
SELECT

    position,

    MAX(height) as Max_Height,

    MIN(height) as Min_Height,

    CAST(AVG(height) as decimal(10,2)) as Avg_Height,

    MAX(weight) as Max_Weight,

    MIN(weight) as Min_Weight,

    CAST(AVG(weight) as decimal(10,2)) as Avg_Weight

FROM player

WHERE

    (position IS NOT NULL)

GROUP BY position

ORDER BY position ASC;
```

```
SQL> SELECT
  2      position,
  3      MAX(height) as Max_Height,
  4      MIN(height) as Min_Height,
  5      CAST(AVG(height) as decimal(10,2)) as Avg_Height,
  6      MAX(weight) as Max_Weight,
  7      MIN(weight) as Min_Weight,
  8      CAST(AVG(weight) as decimal(10,2)) as Avg_Weight
  9  FROM player
 10  WHERE
 11      (position IS NOT NULL)
 12  GROUP BY position
 13  ORDER BY position ASC;

POSIT MAX_HEIGHT MIN_HEIGHT AVG_HEIGHT MAX_WEIGHT MIN_WEIGHT AVG_WEIGHT
----- ---------- ---------- ---------- ---------- ---------- ----------
ATH           76         68      72.08        225        165     190.62
C             79         71      75.04        345        245     290.09
CB            75         61      70.74        220        150     182.15
DB            78         65      71.42        237        146     188.95
DE            81         70      75.15        336        172     250.42
DL            82         68      74.81        380        172     269.95
DS            78         69      72.75        285        168     222.43
DT            79         70       74.5        380        222     289.11
FB            76         65       72.1        308        198     233.76
FL            75         69      72.31        206        159     182.69
FS            75         69      71.81        212        175     196.08
HB            76         69      72.14        250        175     205.21
HOLD          77         75         76        212        187      199.5
ILB           76         69      73.03        264        185     227.95
K             78         62      71.65        260        125     187.77
LB            78         67      73.16        268        160     223.05
LS            80         66      73.26        310        165     223.82
MLB           74         74         74        250        215        235
NG            76         71      73.96        325        245     291.64
NT            77         72      74.31        323        265      293.5
OG            80         71      75.83        395        240     302.51
OL            81         70      76.12        385        220     295.34
OLB           78         69      74.36        265        191     227.53
OT            83         73      77.37        350        240     296.39
P             80         67      73.47        250        150     199.02
PK            78         67      71.72        240        150      189.1
QB            79         69      74.29        257        153     207.18
RB            76         57      70.29        265        135     201.34
ROV           74         70         72        216        188     201.75
S             77         67      72.11        286        159     196.79
SB            76         66      70.12        250        147     192.64
SE            77         70      73.14        208        180     198.86
SN            76         66      73.32        241        185        218
SS            77         70      72.18        218        174     197.42
TB            75         66      70.64        235        150     201.11
TE            81         71      75.97        307        195     241.08
WR            80         61       72.4        263        138        190

37 rows selected.
```

Comparing player jersey duplication frequencies between teams

Description:

This query returns the percentage of a uniform number between the teams. It also displays the count of a particular uniform number, and the total number of uniforms. The same solution can be derived from the following two commands.

Join:

```
SELECT
    a.uniform_number,
    count(a.uniform_number) AS Uniform_Count,
    (SELECT COUNT(a.uniform_number) FROM player) AS Total_Count,
    round(100*(count(a.uniform_number)/(SELECT COUNT(a.uniform_number) FROM
player)), 5) AS Percentage
FROM player a
INNER JOIN player b ON a.player_code = b.player_code AND a.uniform_number IS
NOT NULL
GROUP BY a.uniform_number;
```

```
SQL> SELECT
  2      a.uniform_number,
  3      count(a.uniform_number) AS Uniform_Count,
  4      (SELECT COUNT(a.uniform_number) FROM player) AS Total_Count,
  5      round(100*(count(a.uniform_number)/(SELECT COUNT(a.uniform_number) FROM player)), 5) AS Percentage
  6  FROM player a
  7  INNER JOIN player b ON a.player_code = b.player_code AND a.uniform_number IS NOT NULL
  8  GROUP BY a.uniform_number;

UNI UNIFORM_COUNT TOTAL_COUNT PERCENTAGE
--- ------------- ----------- ----------
58            189       21794     .86721
73            173       21794      .7938
26            235       21794    1.07828
99            201       21794     .92227
55            217       21794     .99569
86            210       21794     .96357
47            213       21794     .97733
1             228       21794    1.04616
45            216       21794      .9911
```

```
25A             1       21794     .00459
85A             1       21794     .00459
13D             1       21794     .00459

139 rows selected.

SQL> _
```

Not all of the rows are shown

Nested Query:

```
SELECT

    uniform_number,

    count(uniform_number) AS Uniform_Count,

    (SELECT COUNT(uniform_number) FROM player) AS Total_Count,

    round(100*(count(uniform_number)/(SELECT COUNT(uniform_number) FROM

player)), 5) AS Percentage

FROM player

WHERE uniform_number IS NOT NULL

GROUP BY uniform_number;
```

```
SQL> SELECT
  2      uniform_number,
  3      count(uniform_number) AS Uniform_Count,
  4      (SELECT COUNT(uniform_number) FROM player) AS Total_Count,
  5      round(100*(count(uniform_number)/(SELECT COUNT(uniform_number) FROM player)), 5) AS Percentage
  6  FROM player
  7  WHERE uniform_number IS NOT NULL
  8  GROUP BY uniform_number;

UNI UNIFORM_COUNT TOTAL_COUNT PERCENTAGE
--- ------------- ----------- ----------
58            189       21276     .88832
73            173       21276     .81312
26            235       21276    1.10453
99            201       21276     .94473
55            217       21276    1.01993
86            210       21276     .98703
47            213       21276    1.00113
1             228       21276    1.07163
45            216       21276    1.01523
61            147       21276     .69092
2B              1       21276      .0047
9D              2       21276      .0094
6D              1       21276      .0047
12D             1       21276      .0047
5D              1       21276      .0047
```

```
4B              1       21276      .0047
25A             1       21276      .0047
85A             1       21276      .0047
13D             1       21276      .0047

139 rows selected.
```

Not all of the rows are shown

<u>Comparing player physical attributes between teams</u>

Description:

This is a complex query where the results are the top or bottom 10 teams that are 'over' or 'below' the average height and weight. This query is used because most teams have little variations within their physical attributes to that point where the data was not significant. The same solution can be derived from the following two commands.

Join:

```
SELECT
    (((AVG(a.height)-(SELECT AVG(height) FROM player))/(SELECT AVG(height)FROM
    player))*100) as hdifperc,
    (((AVG(a.weight)-(SELECT AVG(weight) FROM player))/(SELECT AVG(weight)FROM
    player))*100) as wdifperc,
    (((AVG(a.height)-(SELECT AVG(height) FROM player))/(SELECT AVG(height)FROM
    player))*100)+(((AVG(a.weight)-(SELECT AVG(weight) FROM player))/(SELECT
    AVG(weight) FROM player))*100) as tdifperc,
    b.team_name
FROM player a
INNER JOIN team b ON a.team_code = b.team_code AND b.team_code in (SELECT
team_code FROM player WHERE height IS NOT NULL)
GROUP BY b.team_name
ORDER BY tdifperc ASC;
```

```
SQL> SELECT
  2      (((AVG(a.height)-(SELECT AVG(height) FROM player))/(
  3      (((AVG(a.weight)-(SELECT AVG(weight) FROM player))/(
  4      (((AVG(a.height)-(SELECT AVG(height) FROM player))/(
  5      b.team_name
  6  FROM player a
  7  INNER JOIN team b ON a.team_code = b.team_code AND b.tea
  8  GROUP BY b.team_name
  9  ORDER BY tdifperc ASC;

  HDIFPERC   WDIFPERC   TDIFPERC TEAM_NAME
---------- ---------- ---------- --------------------------
 -.8857082  -5.981176 -6.8668842 Army
-1.1486833 -3.5981018 -4.7467851 Western Michigan
-.66394848 -4.0767938 -4.7407423 Air Force
-.90940855 -3.6197294  -4.529138 Navy
-.19266516  -4.112323 -4.3049881 Arizona
-.65048325 -3.6512449 -4.3017281 Texas Tech
-.85044071 -3.0475298 -3.8979705 New Mexico
.166407848 -3.8517978   -3.68539 Northwestern
-.98716578 -2.4935524 -3.4807182 Troy
-1.0516597 -2.2318258 -3.2834854 Tulane
-.97052066 -2.1633173  -3.133838 Southern Mississippi
```

Not all of the code is shown

```
.401852649 3.02635904 3.42821168 Missouri
 .63748528 3.23039942  3.8678847 Mississippi State
.653513156 3.49112253 4.14463568 Ohio State
1.13889724 3.19229842 4.33119567 Michigan
.852655082 4.26844176 5.12109684 Alabama
1.57428699 4.27414447 5.84843146 Stanford

125 rows selected.

SQL>
```

Not all of the rows are shown

Nested Query:

```
SELECT
    (((AVG(a.height)-(SELECT AVG(height) FROM player))/(SELECT AVG(height)FROM
    player))*100) as hdifperc,
    (((AVG(a.weight)-(SELECT AVG(weight) FROM player))/(SELECT AVG(weight)FROM
    player))*100) as wdifperc,
    (((AVG(a.height)-(SELECT AVG(height) FROM player))/(SELECT AVG(height)FROM
    player))*100)+(((AVG(a.weight)-(SELECT AVG(weight) FROM player))/(SELECT
    AVG(weight) FROM player))*100) as tdifperc,
    b.team_name
FROM player a, team b
WHERE a.team_code = b.team_code AND b.team_code in
    (SELECT team_code FROM player WHERE height IS NOT NULL)
GROUP BY b.team_name
ORDER BY tdifperc ASC;
```

```
SQL> SELECT
  2      (((AVG(a.height)-(SELECT AVG(height) FROM player))/(SELECT AVG(height)
  3      (((AVG(a.weight)-(SELECT AVG(weight) FROM player))/(SELECT AVG(weight)
  4      (((AVG(a.height)-(SELECT AVG(height) FROM player))/(SELECT AVG(height)
  5      b.team_name
  6      FROM player a, team b
  7      WHERE a.team_code = b.team_code AND b.team_code in (SELECT team_code FR
  8      GROUP BY b.team_name
  9      ORDER BY tdifperc ASC;

  HDIFPERC   WDIFPERC   TDIFPERC TEAM_NAME
---------- ---------- ---------- ------------------------------
 -.8857082  -5.981176 -6.8668842 Army
-1.1486833 -3.5981018 -4.7467851 Western Michigan
-.66394848 -4.0767938 -4.7407423 Air Force
-.90940855 -3.6197294  -4.529138 Navy
-.19266516  -4.112323 -4.3049881 Arizona
-.65048325 -3.6512449 -4.3017281 Texas Tech
-.85044071 -3.0475298 -3.8979705 New Mexico
.166407848 -3.8517978   -3.68539 Northwestern
-.98716578 -2.4935524 -3.4807182 Troy
-1.0516597 -2.2318258 -3.2834854 Tulane
-.97052066 -2.1633173  -3.133838 Southern Mississippi
```

Not all of the code is shown

```
.401852649 3.02635904 3.42821168 Missouri
 .63748528 3.23039942  3.8678847 Mississippi State
.653513156 3.49112253 4.14463568 Ohio State
1.13889724 3.19229842 4.33119567 Michigan
.852655082 4.26844176 5.12109684 Alabama
1.57428699 4.27414447 5.84843146 Stanford

125 rows selected.

SQL>
```

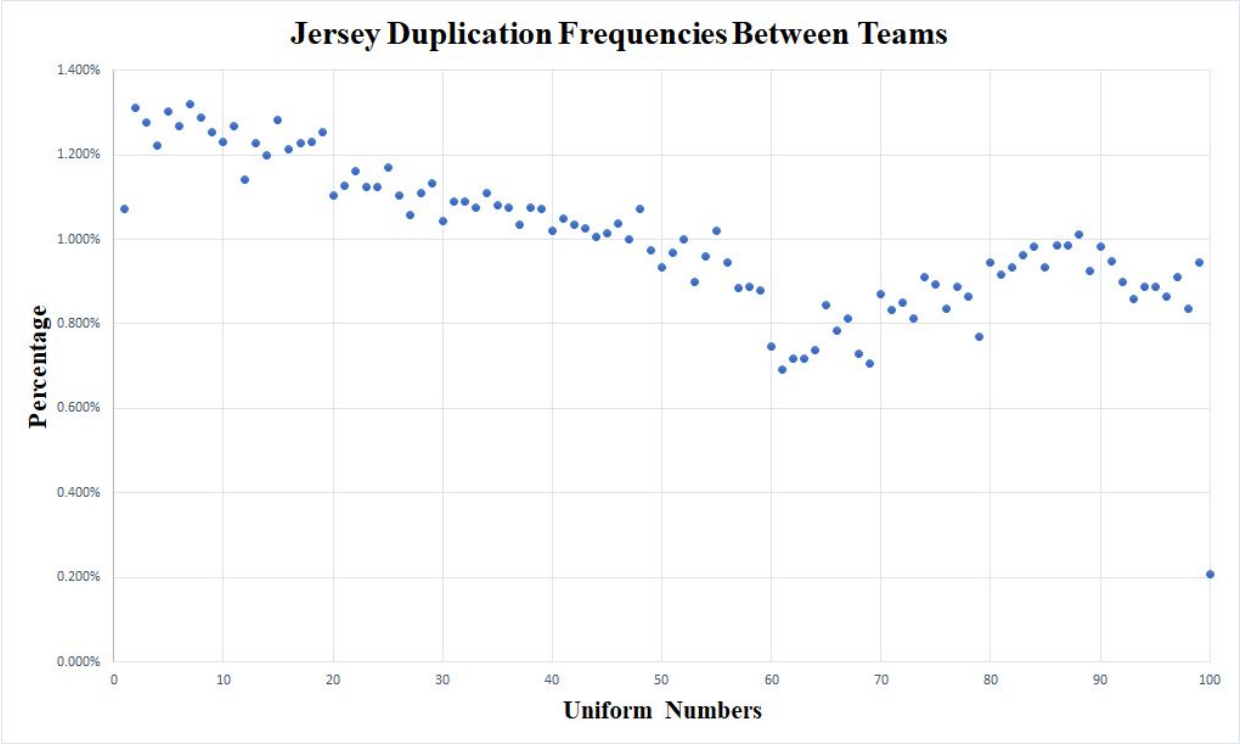
Not all of the rows are shown

Visualization

Comparing height and weight ranges by position

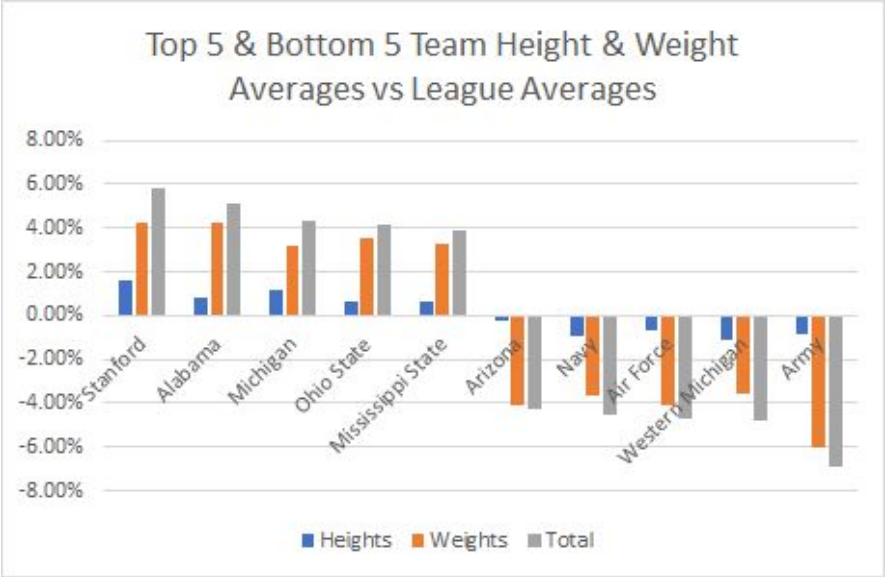
Comparing player height and weight over the first five alphabetically sorted positions.

Comparing player jersey duplication frequencies between teams



Comparing player jersey duplication frequencies between teams on a list sorted ascendingly.

Comparing player attributes between teams



Comparing average player attributes between teams that have the greatest discrepancies.

System Tables

Note: All table descriptions were taken from

https://www.techonthenet.com/oracle/sys_tables/index.php because of their preciseness.

All Tables

Description of relational tables accessible to the user

```
SQL> desc all_tables;
 Name                                                                                                              Null?    Type
 ----------------------------------------------------------------------------------------------------------------- -------- --------------
 OWNER                                                                                                             NOT NULL VARCHAR2(30)
 TABLE_NAME                                                                                                        NOT NULL VARCHAR2(30)
 TABLESPACE_NAME                                                                                                            VARCHAR2(30)
 CLUSTER_NAME                                                                                                               VARCHAR2(30)
 IOT_NAME                                                                                                                   VARCHAR2(30)
 STATUS                                                                                                                     VARCHAR2(8)
 PCT_FREE                                                                                                                   NUMBER
 PCT_USED                                                                                                                   NUMBER
```

```
 DROPPED                                                                                                                    VARCHAR2(3)
 READ_ONLY                                                                                                                  VARCHAR2(3)
 SEGMENT_CREATED                                                                                                            VARCHAR2(3)
 RESULT_CACHE                                                                                                               VARCHAR2(7)

SQL>
```

Not all of the rows are shown

All Constraints

Constraint definitions on accessible tables

```
SQL> desc all_constraints;
 Name                                                                                                           Null?    Type
 -------------------------------------------------------------------------------------------------------------- -------- --------------
 OWNER                                                                                                                   VARCHAR2(120)
 CONSTRAINT_NAME                                                                                                NOT NULL VARCHAR2(30)
 CONSTRAINT_TYPE                                                                                                         VARCHAR2(1)
 TABLE_NAME                                                                                                     NOT NULL VARCHAR2(30)
 SEARCH_CONDITION                                                                                                        LONG
 R_OWNER                                                                                                                 VARCHAR2(120)
 R_CONSTRAINT_NAME                                                                                                       VARCHAR2(30)
 DELETE_RULE                                                                                                             VARCHAR2(9)
 STATUS                                                                                                                  VARCHAR2(8)
 DEFERRABLE                                                                                                              VARCHAR2(14)
 DEFERRED                                                                                                                VARCHAR2(9)
 VALIDATED                                                                                                               VARCHAR2(13)
 GENERATED                                                                                                               VARCHAR2(14)
 BAD                                                                                                                     VARCHAR2(3)
 RELY                                                                                                                    VARCHAR2(4)
 LAST_CHANGE                                                                                                             DATE
 INDEX_OWNER                                                                                                             VARCHAR2(30)
 INDEX_NAME                                                                                                              VARCHAR2(30)
 INVALID                                                                                                                 VARCHAR2(7)
 VIEW_RELATED                                                                                                            VARCHAR2(14)

SQL>
```

Role Tab Privs

Table privileges granted to roles

```
SQL> desc ROLE_TAB_PRIVS;
 Name                                                                                                              Null?    Type
 ----------------------------------------------------------------------------------------------------------------- -------- ------------
 ROLE                                                                                                              NOT NULL VARCHAR2(30)
 OWNER                                                                                                             NOT NULL VARCHAR2(30)
 TABLE_NAME                                                                                                        NOT NULL VARCHAR2(30)
 COLUMN_NAME                                                                                                                VARCHAR2(30)
 PRIVILEGE                                                                                                         NOT NULL VARCHAR2(40)
 GRANTABLE                                                                                                                  VARCHAR2(3)

SQL> _
```

All Users

Information about all users of the database

```
SQL> desc all_users;
 Name                                                                                                              Null?    Type
 ----------------------------------------------------------------------------------------------------------------- -------- ------------
 USERNAME                                                                                                          NOT NULL VARCHAR2(30)
 USER_ID                                                                                                           NOT NULL NUMBER
 CREATED                                                                                                           NOT NULL DATE
```

All Types

Description of types accessible to the user

```
SQL> desc ALL_TYPES;
 Name                                                                                                              Null?    Type
 ----------------------------------------------------------------------------------------------------------------- -------- ------------
 OWNER                                                                                                                      VARCHAR2(30)
 TYPE_NAME                                                                                                                  VARCHAR2(30)
 TYPE_OID                                                                                                                   RAW(16)
 TYPECODE                                                                                                                   VARCHAR2(30)
 ATTRIBUTES                                                                                                                 NUMBER
 METHODS                                                                                                                    NUMBER
 PREDEFINED                                                                                                                 VARCHAR2(3)
 INCOMPLETE                                                                                                                 VARCHAR2(3)
 FINAL                                                                                                                      VARCHAR2(3)
 INSTANTIABLE                                                                                                               VARCHAR2(3)
 SUPERTYPE_OWNER                                                                                                            VARCHAR2(30)
 SUPERTYPE_NAME                                                                                                             VARCHAR2(30)
 LOCAL_ATTRIBUTES                                                                                                           NUMBER
 LOCAL_METHODS                                                                                                              NUMBER
 TYPEID                                                                                                                     RAW(16)
```

All Catalog

All tables, views, synonyms, sequences accessible to the user

```
SQL> desc ALL_CATALOG;
 Name                                                                                                              Null?    Type
 ----------------------------------------------------------------------------------------------------------------- -------- ------------
 OWNER                                                                                                             NOT NULL VARCHAR2(30)
 TABLE_NAME                                                                                                        NOT NULL VARCHAR2(30)
 TABLE_TYPE                                                                                                                 VARCHAR2(11)
```

Schema Constraints

Restrict Constraint

A restrict constraint in Oracle can be set with a create or alter table statement on references can apply to either update or delete operations for the table specified. When a value in a column in the parent table's primary key is updated and a restrict constraint is specified then Oracle checks for any foreign key dependencies and denies the update if the value updated exists as a foreign dependency. Similarly a delete statement on a table with a restrict constraint will be rejected if there are any foreign dependencies tied to the primary keys of the requested deleted records.

BEFORE

Added new records with dependencies for test.

Parent table conference with primary key conference_code (1001):

```
SQL> insert into conference values (1001, 'Unrestricted Conference', 'FBS');

1 row created.
```

Child table team with foreign key dependency on conference_code (1001):

```
SQL> insert into team values (3001, 'An unrestricted team', 1001);

1 row created.
```

Attempt to update conference_code value:

```
SQL> update conference set conference_code = 1002 where conference_code = 1001;
update conference set conference_code = 1002 where conference_code = 1001
*
ERROR at line 1:
ORA-02292: integrity constraint (USER101.SYS_C007043) violated - child record
found
```

Attempt to delete conference_code value:

```
SQL> delete from conference where conference_code = 1001;
delete from conference where conference_code = 1001
*
ERROR at line 1:
ORA-02292: integrity constraint (USER101.SYS_C007043) violated - child record
found
```

AFTER

All Oracle foreign key constraints implicitly enable a restrict constraint. Whereas other SQL languages include the 'restrict' keyword Oracle does not:

```
SQL> alter table team add constraint fk_restrict foreign key (conference_code) references conference(conference_code) on delete restrict on update restrict;
alter table team add constraint fk_restrict foreign key (conference_code) references conference(conference_code) on delete restrict on update restrict
                                                                                                                                                *
ERROR at line 1:
ORA-00905: missing keyword
```

As you can see below from the DBMS metadata, the team table includes the foreign key constraint that implicitly includes an on delete and on update restrict constraint.

```
SQL> set long 10000;
SQL> select dbms_metadata.get_ddl( 'TABLE', 'TEAM', 'USER101' ) from dual;

DBMS_METADATA.GET_DDL('TABLE','TEAM','USER101')
-------------------------------------------------------------------------------

  CREATE TABLE "USER101"."TEAM"
   (    "TEAM_CODE" NUMBER,
        "TEAM_NAME" VARCHAR2(30),
        "CONFERENCE_CODE" NUMBER NOT NULL ENABLE,
         PRIMARY KEY ("TEAM_CODE")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DE
FAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "SYSTEM"  ENABLE,

DBMS_METADATA.GET_DDL('TABLE','TEAM','USER101')
-------------------------------------------------------------------------------
         UNIQUE ("TEAM_NAME")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DE
FAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "SYSTEM"  ENABLE,
         FOREIGN KEY ("CONFERENCE_CODE")
          REFERENCES "USER101"."CONFERENCE" ("CONFERENCE_CODE") ENABLE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

DBMS_METADATA.GET_DDL('TABLE','TEAM','USER101')
-------------------------------------------------------------------------------
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DE
FAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "SYSTEM"
```

Cascade Delete

As a workaround to the foreign key constraint, Oracle SQL allows a delete on a parent record if the cascade delete constraint is added to a foreign key reference. On delete Oracle will remove all children records that have a foreign key referencing the parent record.

BEFORE

Prior to implementing an on delete cascade modification to the foreign key constraint, we continue to use the data inserted above for both team and conference.

Parent table conference with primary key conference_code (1001):

```
SQL> insert into conference values (1001, 'Unrestricted Conference', 'FBS');

1 row created.
```

Child table team with foreign key dependency on conference_code (1001):

```
SQL> insert into team values (3001, 'An unrestricted team', 1001);

1 row created.
```

Attempt to delete conference_code value:

```
SQL> delete from conference where conference_code = 1001;
delete from conference where conference_code = 1001
*
ERROR at line 1:
ORA-02292: integrity constraint (USER101.SYS_C007043) violated - child record
found
```

To update foreign key when a constraint name was not specified we must check the system's metadata to see what the constraint was automatically named.

SELECT a.table_name, a.column_name, a.constraint_name FROM ALL_CONS_COLUMNS A, ALL_CONSTRAINTS C where A.CONSTRAINT_NAME = C.CONSTRAINT_NAME and C.CONSTRAINT_TYPE = 'R' and C.TABLE_NAME='TEAM';

```
TEAM
CONFERENCE_CODE
SYS_C007043
```

AFTER

Now that we have the foreign key constraint name automatically generated by the system we can delete this constraint:

```
SQL> alter table team drop constraint SYS_C007043;

Table altered.
```

We now add the constraint again but this time with the on delete cascade condition.

```
SQL> alter table team add constraint fk_cascade foreign key (conference_code) references conference(conference_code) on delete cascade enable;

Table altered.
```

We can verify this change took place in the DBMS metadata:

```
SQL> select dbms_metadata.get_ddl( 'TABLE', 'TEAM', 'USER101' ) from dual;

DBMS_METADATA.GET_DDL('TABLE','TEAM','USER101')
-------------------------------------------------------------------------------

  CREATE TABLE "USER101"."TEAM"
   (    "TEAM_CODE" NUMBER,
        "TEAM_NAME" VARCHAR2(30),
        "CONFERENCE_CODE" NUMBER NOT NULL ENABLE,
         PRIMARY KEY ("TEAM_CODE")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DE
FAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "SYSTEM"  ENABLE,

DBMS_METADATA.GET_DDL('TABLE','TEAM','USER101')
-------------------------------------------------------------------------------
         UNIQUE ("TEAM_NAME")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DE
FAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "SYSTEM"  ENABLE,
         CONSTRAINT "FK_CASCADE" FOREIGN KEY ("CONFERENCE_CODE")
          REFERENCES "USER101"."CONFERENCE" ("CONFERENCE_CODE") ON DELETE CASCADE ENA
E
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING

DBMS_METADATA.GET_DDL('TABLE','TEAM','USER101')
-------------------------------------------------------------------------------
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DE
FAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "SYSTEM"
```

We can now attempt to delete the test conference record that was connected with our test team record.

```
SQL> delete from conference where conference_code = 1001;

1 row deleted.
```

We can then verify that the cascade delete was applied to the referenced team table. There was 1 record (team_code 3001 referencing conference_code 1001) and now it is gone.

```
SQL> select * from team where conference_code = 1001;

no rows selected
```

Set Default

In Oracle a default constraint will automatically set a value to all previous records when adding a new column. This is especially useful when adding a new column to tables with existing data if you want to also add the not null constraint, as the new column would not normally be allowed since the existing records do not have a value for the new column. Adding the default constraint populates the previous columns and satisfies the not null requirement.

BEFORE

First let's look at adding a new column with neither default nor not null constraints.

```
SQL> alter table conference add beforetest varchar(30);

Table altered.
```

We are able to add the column and existing tables have a value of null.

```
SQL> select * from conference where conference_code = 2001;

CONFERENCE_CODE CONFERENCE_NAME                SUB BEFORETEST
--------------- ------------------------------ --- ------------------------------
           2001 Sample Conference              FCS
```

Now let's drop the same field and try to add it again using the not null constraint.

```
SQL> alter table conference drop column beforetest;

Table altered.

SQL> alter table conference add beforetest varchar(30) not null;
alter table conference add beforetest varchar(30) not null
            *
ERROR at line 1:
ORA-01758: table must be empty to add mandatory (NOT NULL) column
```

As you can see the column cannot be added because a not null column add without a default must be empty.

AFTER

We can now try to add the new column as not null to the existing non empty table conference by adding the default constraint.

```
SQL> alter table conference add aftertest varchar(30) default 'Default Value Test' not null;

Table altered.
```

The column is added successfully,

If we check our existing records we can see the default value was inserted.

```
SQL> select * from conference where conference_code = 2001;

CONFERENCE_CODE CONFERENCE_NAME                SUB AFTERTEST
--------------- ------------------------------ --- ------------------------------
           2001 Sample Conference              FCS Default Value Test
```

Oracle Sequence Object

```
SQL> CREATE SEQUENCE team_sq
  2  START WITH 3000
  3  INCREMENT BY 1
  4  MAXVALUE 3500
  5  NOCYCLE
  6  CACHE 20;

Sequence created.

SQL>
```

Creating a sequence called team_sq that starts at 3000 and is incremented by 1 every time it is referenced.

```
SQL> SELECT * FROM team WHERE rownum <= 10 ORDER BY team_code DESC;

 TEAM_CODE TEAM_NAME                      CONFERENCE_CODE
---------- ------------------------------ ---------------
      2915 Wofford                                    912
      2678 Arkansas-Pine Bluff                        916
      1320 Presbyterian                               826
      1092 Gardner-Webb                               826
      1068 Elon                                       912
      1004 Central Arkansas                           914
       817 Youngstown State                           853
       813 Yale                                       865
       811 Wyoming                                   5486
       796 Wisconsin                                  827

10 rows selected.

SQL>
```

Displaying 10 rows from TABLE team in descending order to provide as a BEFORE example.

```
SQL> INSERT INTO team (team_code, team_name, conference_code)
  2  VALUES (team_sq.NEXTVAL, 'Chico State', 916);

1 row created.

SQL> INSERT INTO team (team_code, team_name, conference_code)
  2  VALUES (team_sq.NEXTVAL, 'Butte College', 827);

1 row created.

SQL> INSERT INTO team (team_code, team_name, conference_code)
  2  VALUES (team_sq.NEXTVAL, 'Yuba College', 912);

1 row created.
```

Inserting values into the TEAM table using the sequence team_sq in place of unique team codes.

```
SQL> SELECT * FROM team WHERE rownum <= 10 ORDER BY team_code DESC;

 TEAM_CODE TEAM_NAME                      CONFERENCE_CODE
---------- ------------------------------ ---------------
      3002 Yuba College                               912
      3001 Butte College                              827
      3000 Chico State                                916
      2915 Wofford                                    912
      2678 Arkansas-Pine Bluff                        916
      1320 Presbyterian                               826
      1092 Gardner-Webb                               826
      1068 Elon                                       912
      1004 Central Arkansas                           914
       817 Youngstown State                           853

10 rows selected.

SQL>
```

Displaying 10 rows from TABLE team in descending order to show how the three topmost entries have team_codes that were generated from the sequence team_sq.

Layered Design with Views

arizona_state View

```
CREATE VIEW arizona_state AS
SELECT
    a.first_name, a.last_name, a.uniform_number, a.height, a.weight, b. team_code, b.team_name
  4  FROM player a, team b
  5  WHERE a.team_code = 28 AND a.team_code = b.team_code;

View created.

SQL> descr arizona_state
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 FIRST_NAME                                NOT NULL VARCHAR2(30)
 LAST_NAME                                 NOT NULL VARCHAR2(30)
 UNIFORM_NUMBER                                     VARCHAR2(3)
 HEIGHT                                             NUMBER
 WEIGHT                                             NUMBER
 TEAM_CODE                                 NOT NULL NUMBER
 TEAM_NAME                                          VARCHAR2(30)

SQL>
```

Creating a view called arizona_state that has the first names, last names, uniform numbers, height, weight, team codes, and team names.

```
SQL> SELECT * FROM arizona_state
  2  WHERE rownum <= 10;

FIRST_NAME                     LAST_NAME                      UNI     HEIGHT     WEIGHT  TEAM_CODE TEAM_NAME
------------------------------ ------------------------------ --- ---------- ---------- ---------- ---------------
Alonzo                         Agwuenu                        4           76        207         28 Arizona State
Sil                            Ajawara                        58          76        299         28 Arizona State
Dante                          Alexander                      37          69        191         28 Arizona State
Marcus                         Ball                           10          75        205         28 Arizona State
Mike                           Bercovici                      2           73        196         28 Arizona State
Ezekiel                        Bishop                         23          71        199         28 Arizona State
Carl                           Bradford                       52          73        243         28 Arizona State
Dwain                          Bradshaw                       22          71        218         28 Arizona State
Alex                           Bykovskiy                      48          73        218         28 Arizona State
Lloyd                          Carrington                     17          71        188         28 Arizona State

10 rows selected.

SQL>
```

Displaying 10 rows from arizona_state using CREATE VIEW.

arizona_state_conference View

```
SQL> CREATE VIEW arizona_state_conference AS
  2  SELECT
  3      a.conference_name, a.sub_division, b.team_code, b.team_name
  4  FROM conference a, team b
  5  WHERE b.team_code = 28 AND b.conference_code = a.conference_code;

View created.

SQL>
```

Creating a view called arizona_state_conference that has the conference name, sub division, team code, and the team name of Arizona State.

```
SQL> SELECT* from arizona_state_conference;

CONFERENCE_NAME                SUB  TEAM_CODE TEAM_NAME
------------------------------ --- ---------- ------------------------------
Pac-12 Conference              FBS         28 Arizona State

SQL>
```

Describing the contents of arizona_state_conference.

tall_arizona View

```
SQL> CREATE VIEW tall_arizona AS
  2  SELECT
  3      first_name, last_name, height
  4  FROM arizona_state
  5  WHERE rownum <= 10
  6  ORDER BY height DESC;

View created.

SQL>
```

Creating a view called tall_arizona that has the first name, last name, and the height of the tallest ten players in the team.

```
SQL> select *
  2  from tall_arizona;

FIRST_NAME                     LAST_NAME                        HEIGHT
------------------------------ ------------------------------ ----------
Alonzo                         Agwuenu                                76
Sil                            Ajawara                                76
Marcus                         Ball                                   75
Mike                           Bercovici                              73
Alex                           Bykovskiy                              73
Carl                           Bradford                               73
Dwain                          Bradshaw                               71
Ezekiel                        Bishop                                 71
Lloyd                          Carrington                             71
Dante                          Alexander                              69

10 rows selected.

SQL>
```

Describing the contents of tall_arizona.

short_arizona View

```
SQL> CREATE VIEW short_arizona AS
  2  SELECT
  3      first_name, last_name, height
  4  FROM arizona_state
  5  WHERE rownum <= 10
  6  ORDER BY height;

View created.
```

Creating a view called short_arizona that has the first name, last name, and the height of the shortest ten players in the team.

```
SQL> select *
  2  from short_arizona;

FIRST_NAME                     LAST_NAME                        HEIGHT
------------------------------ ------------------------------ ----------
Dante                          Alexander                              69
Dwain                          Bradshaw                               71
Ezekiel                        Bishop                                 71
Lloyd                          Carrington                             71
Mike                           Bercovici                              73
Alex                           Bykovskiy                              73
Carl                           Bradford                               73
Marcus                         Ball                                   75
Alonzo                         Agwuenu                                76
Sil                            Ajawara                                76

10 rows selected.

SQL> _
```

Describing the contents of short_arizona.

arizona_playconf View

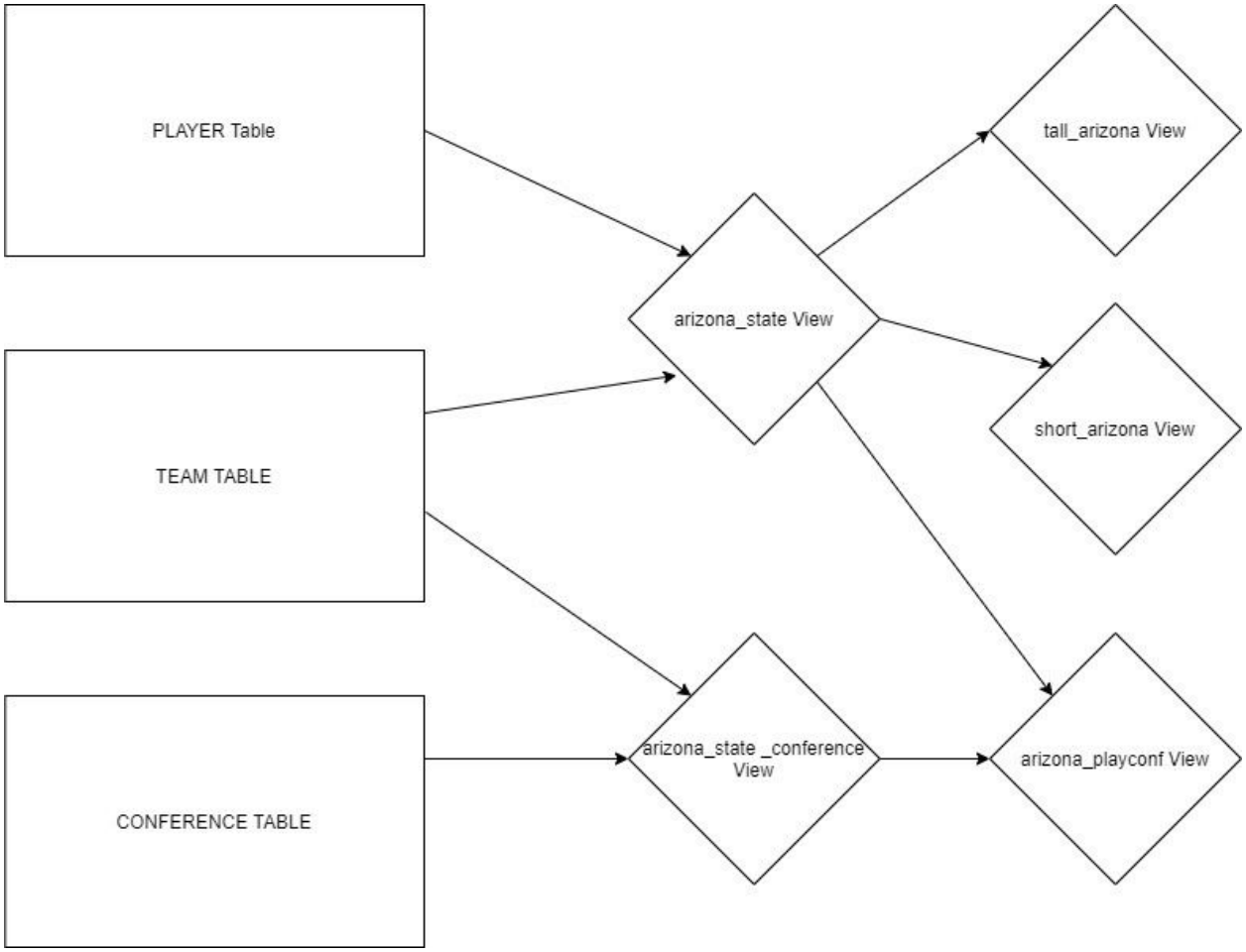
```
SQL> CREATE VIEW arizona_playconf AS
  2  SELECT
  3      a.first_name, a.last_name, b.conference_name
  4  FROM arizona_state a, arizona_state_conference b;

View created.

SQL>
```

Creating a view called arizona_playconf that has the first name, last name, and the conference name of the players in Arizona State.

```
SQL> select *
  2  from arizona_playconf
  3  WHERE rownum <=10;

FIRST_NAME                     LAST_NAME                      CONFERENCE_NAME
------------------------------ ------------------------------ ------------------
Alonzo                         Agwuenu                        Pac-12 Conference
Sil                            Ajawara                        Pac-12 Conference
Dante                          Alexander                      Pac-12 Conference
Marcus                         Ball                           Pac-12 Conference
Mike                           Bercovici                      Pac-12 Conference
Ezekiel                        Bishop                         Pac-12 Conference
Carl                           Bradford                       Pac-12 Conference
Dwain                          Bradshaw                       Pac-12 Conference
Alex                           Bykovskiy                      Pac-12 Conference
Lloyd                          Carrington                     Pac-12 Conference

10 rows selected.

SQL>
```

Describing the contents of arizona_playconf by displaying 10 rows.

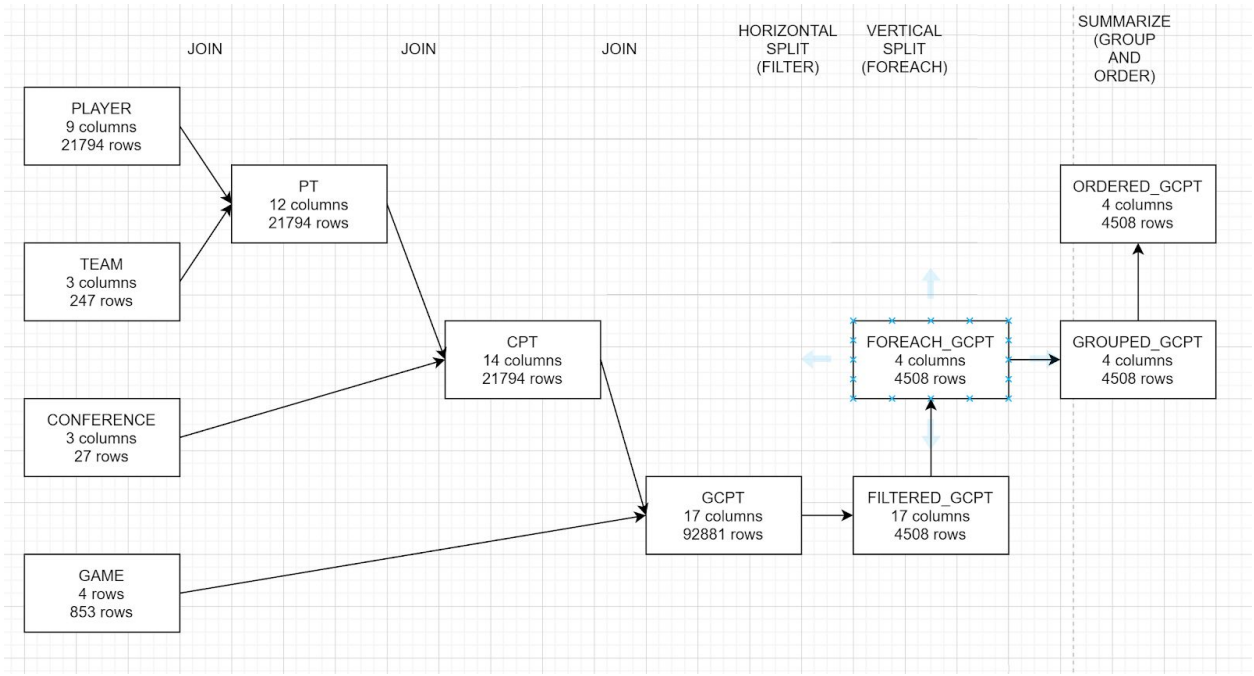
Diagram


PLAYER Table
TEAM TABLE
CONFERENCE TABLE
arizona_state View
tall_arizona View
short_arizona View
arizona_state _conference View
arizona_playconf View

Pig on Hadoop

We devised a scenario where the Independent NCAA football conference asks us for a list of their players who traveled during the season along with the days they traveled. Their interest is for potential contact tracing investigations. We then build a pig script ('indytravelers.pig') to query our combination of datasets we have for players, games, teams and conferences to find all away player rosters as they would be traveling away from their school.

The diagram below describes our 'indytravelers.pig' script and includes the statistics for the columns and rows returned at each step. Following the diagram is the actual script used.




```
/* LOAD OF PLAYER DATASET */
player = LOAD '2013player.csv' USING PigStorage(',') AS (player_code:int,
team_code:int, last_name: chararray, first_name: chararray, uniform_number:
int, class: chararray, position: chararray, height:int, weight:int);
STORE player INTO '/user/root/p4z/player';
cnt_player = FOREACH (GROUP player ALL) GENERATE COUNT(player);
```

```
STORE cnt_player INTO '/user/root/p4z/cnt_player';

/* LOAD OF TEAM DATASET */

team = LOAD '2013team.csv' USING PigStorage(',') AS (team_code:int,

team_name:chararray, conference_code:int);

STORE team INTO '/user/root/p4z/team';

cnt_team = FOREACH (GROUP team ALL) GENERATE COUNT(team);

STORE cnt_team INTO '/user/root/p4z/cnt_team';


/* JOIN OF PLAYER AND TEAM (PT) */

pt = JOIN player BY team_code, team BY team_code;

STORE pt INTO '/user/root/p4z/pt';

cnt_pt = FOREACH (GROUP pt ALL) GENERATE COUNT(pt);

STORE cnt_pt INTO '/user/root/p4z/cnt_pt';


/* LOAD OF CONFERENCE DATASET */

conference = LOAD '2013conference.csv' USING PigStorage(',') AS (code:int,

conference_name:chararray, sub_division:chararray);

STORE conference INTO '/user/root/p4z/conference';

cnt_conference = FOREACH (GROUP conference ALL) GENERATE COUNT(conference);

STORE cnt_conference INTO '/user/root/p4z/cnt_conference';


/* JOIN OF CONFERENCE WITH PLAYER AND TEAM (CPT) */

cpt = JOIN pt BY code, conference BY conference_code;

STORE cpt INTO '/user/root/p4z/cpt';

cnt_cpt = FOREACH (GROUP cpt ALL) GENERATE COUNT(cpt);

STORE cnt_cpt INTO '/user/root/p4z/cnt_cpt';
```

```
/* LOAD OF GAME DATASET */
game = LOAD '2013game.csv' USING PigStorage(',') AS (game_code:int,
game_date:chararray, home_code:int, away_code:int);
STORE game INTO '/user/root/p4z/game';
cnt_game = FOREACH (GROUP game ALL) GENERATE COUNT(game);
STORE cnt_game INTO '/user/root/p4z/cnt_game';

/* JOIN OF GAME WITH CONFERENCE AND PLAYER AND TEAM (GCPT) */
gcpt = JOIN game BY away_code, cpt BY team_code;
STORE gcpt INTO '/user/root/p4z/gcpt';
cnt_gcpt = FOREACH (GROUP gcpt ALL) GENERATE COUNT(gcpt);
STORE cnt_gcpt INTO '/user/root/p4z/cnt_gcpt';

/* HORIZONTAL SPLIT OF GAME CONFERENCE PLAYER TEAM */
filter_gcpt = FILTER gcpt BY conference_name == 'Independent';
STORE filter_gcpt INTO '/user/root/p4z/filter_gcpt';
cnt_filter_gcpt = FOREACH (GROUP filter_gcpt ALL) GENERATE COUNT(filter_gcpt);
STORE cnt_filter_gcpt INTO '/user/root/p4z/cnt_filter_gcpt';

/* VERTICAL SPLIT OF GAME CONFERENCE PLAYER TEAM */
foreach_gcpt = FOREACH filter_gcpt GENERATE game_date, team_name, last_name,
first_name;
STORE foreach_gcpt INTO '/user/root/p4z/foreach_gcpt';
cnt_foreach_gcpt = FOREACH (GROUP foreach_gcpt ALL) GENERATE
COUNT(foreach_gcpt);
STORE cnt_foreach_gcpt INTO '/user/root/p4z/cnt_foreach_gcpt';
```

```
/* GROUPING OF GAME CONFERENCE PLAYER TEAM */
grouped_gcpt = GROUP filter_gcpt BY (last_name, first_name, team_name);
STORE grouped_gcpt INTO '/user/root/p4z/grouped_gcpt';
cnt_grouped_gcpt = FOREACH (GROUP grouped_gcpt ALL) GENERATE
COUNT(grouped_gcpt);
STORE cnt_grouped_gcpt INTO '/user/root/p4z/cnt_grouped_gcpt';

/* ORDERING OF GAME CONFERENCE PLAYER TEAM */
ordered_gcpt = ORDER grouped_gcpt BY (last_date, first_name, team_name,
game_date);
STORE ordered_gcpt INTO '/user/root/p4z/ordered_gcpt';
cnt_ordered_gcpt = FOREACH (GROUP ordered_gcpt ALL) GENERATE
COUNT(ordered_gcpt);
STORE cnt_ordered_gcpt INTO '/user/root/p4z/cnt_ordered_gcpt';
```

Store Procedure

In shell we created a .sql file creating the stored procedure “schedule_games”:

1.) Takes input parameters: The procedure takes in a start date, two team codes and a number representing how many games to schedule.

2.) Loops: The procedure runs up to the number of games requested in the input parameter up to a maximum of 5 times.

3.) Multiple record access: The procedure both selects the latest game code so that it can assign the next available key value and also inserts multiple values into the game table.

4.) Additional Logic: The procedure schedules games on the same day each subsequent week from the start date given and alternates between home and away teams on each new game schedule record.

```
ubuntu@ip-172-31-20-133:~$ cat sp_example.sql
create or replace procedure schedule_games (start_date date, team1 number, team2 number, num_games number) is
v_gdate date;
v_at number;
v_bt number;
v_teamtmp number;
v_num number;
v_code number;
begin
  v_gdate := start_date;
  v_at := team1;
  v_bt := team2;
  v_num := num_games;
  if v_num > 5 then
    v_num := 5;
  end if;
  select game_code into v_code
  from game
  where rownum = 1
  order by game_code desc;
  for i in 1..v_num loop
    v_code := v_code + 1;
    insert into game values (v_code, v_gdate, v_at, v_bt);
    v_teamtmp := v_at;
    v_at := v_bt;
    v_bt := v_teamtmp;
    v_gdate := v_gdate+7;
  end loop;
end;
/
ubuntu@ip-172-31-20-133:~$
```

After creating the sql file we log into sqlplus and run it using the @ symbol. This creates our procedure.

```
SQL> @sp_example;

Procedure created.

SQL>
```

After the procedure is created we execute it using the exec command. We pass parameters including an optional ANSI date literal to ensure our dates convert properly.

```
SQL> exec schedule_games(date '2020-06-27', 5, 8, 5);

PL/SQL procedure successfully completed.

SQL>
```

By checking the table we can see the records that were automatically generated by our stored procedure.

```
SQL> select to_char(game_code) as code, game_date, home_team_code, visit_team_code from game where game_date >= date '2020-06-27';

CODE                                   GAME_DATE HOME_TEAM_CODE VISIT_TEAM_CODE
-------------------------------------- --------- -------------- ---------------
2915005120130832                       27-JUN-20              8               5
2915005120130833                       04-JUL-20              5               8
2915005120130834                       11-JUL-20              8               5
2915005120130835                       18-JUL-20              5               8
2915005120130836                       25-JUL-20              8               5

SQL>
```

PL/SQL Cursor

In Oracle cursors can be implicit or explicit. Explicit cursors are named, and for our example we created one for a weight report on our NCAA football database. We pass two arguments to the procedure: a low weight and a high weight. The procedure then sets up a cursor pointing to a select statement using these two input parameters in the where clause. The report itself then iterates through each record and prints out a weight header field whenever a new weight is encountered. It then prints a team name header if a new header is encountered. Finally every player that matches the criteria has their last and first name printed out.

```
ubuntu@ip-172-31-20-133: $ cat tw1.sql
create or replace procedure weight_report (low_weight player.weight%type, high_weight player.weight%type) is
v_ln player.last_name%type;
v_fn player.first_name%type;
v_w1 player.weight%type;
v_w2 player.weight%type;
v_tn team.team_name%type;
v_tn2 team.team_name%type;
cursor c_player is
   select last_name, first_name, weight, team_name from player a, team b where a.team_code = b.team_code and weight >= low_weight and weight <= high_weight order by weight asc, team_name, last_name;
begin
   dbms_output.put_line('Weight Report NCAA Football Players between ' || low_weight || ' and ' || high_weight || ' lbs');
   open c_player;
   v_w2 := 0;
   v_tn2 := ' ';
   loop
   fetch c_player into v_ln, v_fn, v_w1, v_tn;
      exit when c_player%notfound;
          if v_w1 != v_w2 then
             dbms_output.put_line( CHR(13) || CHR(10) || v_w1 || ' LBS: ');
          end if;
          if v_tn != v_tn2 then
             dbms_output.put_line(CHR(13) || CHR(10) || 'Team ' || v_tn);
          end if;
      dbms_output.put_line('Player ' || v_ln || ', ' || v_fn);
          v_w2 := v_w1;
   end loop;
   close c_player;
end;
/
```

In SQL*Plus we create the procedure by running the .sql file with the @ command.

```
SQL> @tw1

Procedure created.

SQL>
```

Now we can activate our procedure that contains our player cursor by calling the exec command on our stored procedure's name.

```
SQL> exec weight_report(350,355);
Weight Report NCAA Football Players between 350 and 355 lbs

350 LBS:

Team California
Player Cochran, Aaron

Team Clemson
Player Region, Spencer

Team Kentucky
Player Gruenschlaeger, John

Team Louisville
Player Brown, Jamon

Team TCU
Player Pryor, Matt

Team UNLV
Player Alvarez, Bobby

Team UTEP
Player Silvas, Josh

Team UTSA
Player Owens, Breyun

351 LBS:

Team Tennessee
Player McCullers, Daniel

355 LBS:

Team Georgia Tech
Player Devine, Shamire

Team Notre Dame
Player Carr, Kevin

PL/SQL procedure successfully completed.

SQL>
```

PL/SQL trigger

Triggers allow us to add additional programming logic to execute upon a given condition within the DBMS. For our database we added a DML trigger to occur before any insert or update statements to implement a complex input validation routine that could not be expressed through a standard integrity constraint. Each player has a position attribute and uniform number, and the NCAA guidelines specify that certain positions should only have specific uniform numbers.

We start by creating a .sql file as displayed below:

```
ubuntu@ip-172-31-20-133: $ cat j1.sql
create or replace trigger jersey_formats
before insert or update of uniform_number, position on player
for each row
begin
if (:new.uniform_number > 50 and :new.position like '%B') then
  raise_application_error(-20000,'Backs must have Uniform number between 1-49');
end if;
if ((:new.uniform_number < 50 or :new.uniform_number > 59) and :new.position like '%C') then
  raise_application_error(-20000,'Centers must have Uniform number between 50-59');
end if;
if ((:new.uniform_number < 60 or :new.uniform_number > 69) and :new.position like '%G') then
  raise_application_error(-20000,'Guards must have Uniform number between 60-69');
end if;
if ((:new.uniform_number < 70 or :new.uniform_number > 79) and :new.position like '%T') then
  raise_application_error(-20000,'Tackles must have Uniform number between 70-79');
end if;
if (:new.uniform_number < 80 and :new.position like '%E') then
  raise_application_error(-20000,'Ends must have Uniform number between 80-99');
end if;
if (:new.uniform_number < 1 or :new.uniform_number > 99) then
  raise_application_error(-20000,'Uniform number must be between 1-99');
end if;
end;
/
ubuntu@ip-172-31-20-133: $
```

As part of our input validation we will not allow positions with QB to have a uniform number greater than 50.

Before we implement our trigger, we test to see that normally any valid number can be entered into the position field. In this test case even though the position is 'QB' there is no logic to stop the uniform from being set to 88.

```
SQL> update player set uniform_number = 88 where position = 'QB' and rownum = 1;

1 row updated.

SQL>
```

After running the command above we validate that the data is updated.

```
SQL> select uniform_number, position from player where position = 'QB' and rownum = 1;

UNI POSIT
--- -----
88  QB
```

Now we will implement our new trigger by running the .sql file using the @ command.

```
SQL> @j1.sql;

Trigger created.

SQL>
```

The trigger is set. We try again to update a player with 'QB' position to a uniform number greater than 50. We correctly receive the error message specified in our trigger logic.

```
SQL> update player set uniform_number = 83 where position = 'QB' and rownum = 1;
update player set uniform_number = 83 where position = 'QB' and rownum = 1
       *
ERROR at line 1:
ORA-20000: Backs must have Uniform number between 1-49
ORA-06512: at "USER101.JERSEY_FORMATS", line 3
ORA-04088: error during execution of trigger 'USER101.JERSEY_FORMATS'


SQL>
```

Finally we check that a normal update can take place once satisfy the trigger validation.

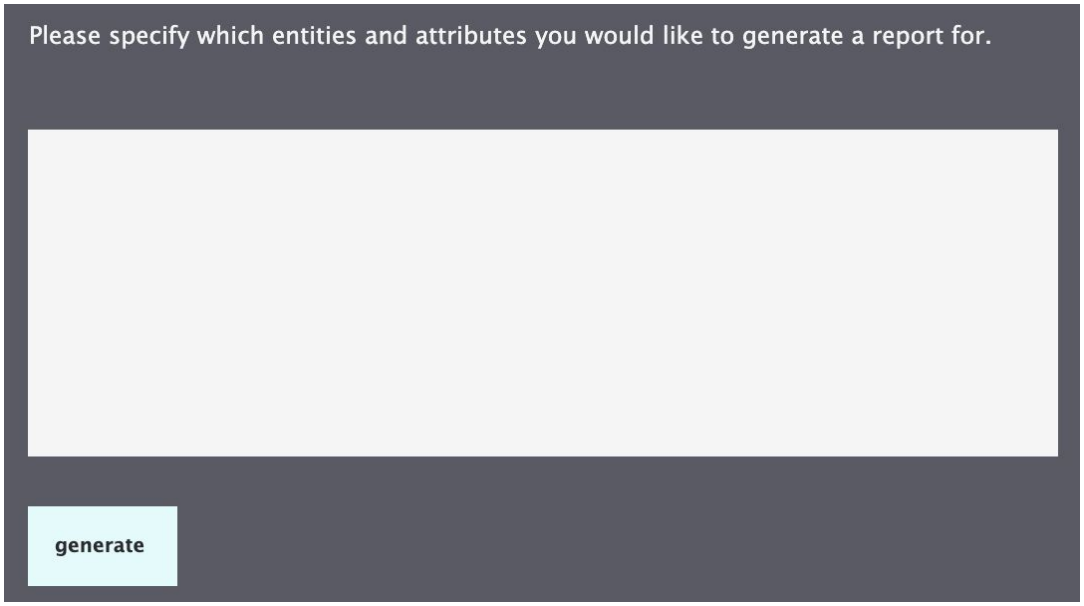
```
SQL> update player set uniform_number = 15 where position = 'QB' and rownum = 1;

1 row updated.
```

And validate that the data is updated properly.

```
SQL> select uniform_number, position from player where position = 'QB' and rownum = 1;

UNI POSIT
--- -----
15  QB
```

Live User Interface

To link our live UI to SQL backend, we used JavaScript and Express framework.



Our function call to the backend returns the data in json format.


```
[
  RowDataPacket {
    idteam: 2,
    teamName: 'Abilene Christian',
    'conference code': 99005
  },
  RowDataPacket {
    idteam: 5,
    teamName: 'Akron',
    'conference code': 875
  },
  RowDataPacket {
    idteam: 6,
    teamName: 'Alabama A&M',
    'conference code': 916
  },
  RowDataPacket {
    idteam: 7,
    teamName: 'Alabama St.',
    'conference code': 916
  },
  RowDataPacket {
    idteam: 8,
    teamName: 'Alabama',
    'conference code': 911
```

Addendum

Hurdles

- There was an error in trying to filter the TEAM table and the CONFERENCE table in PIG after joining them together. Initially, the code was:

  merge2 = FILTER merge BY (team.team_code <= 700);

  Dumping that would produce the error:

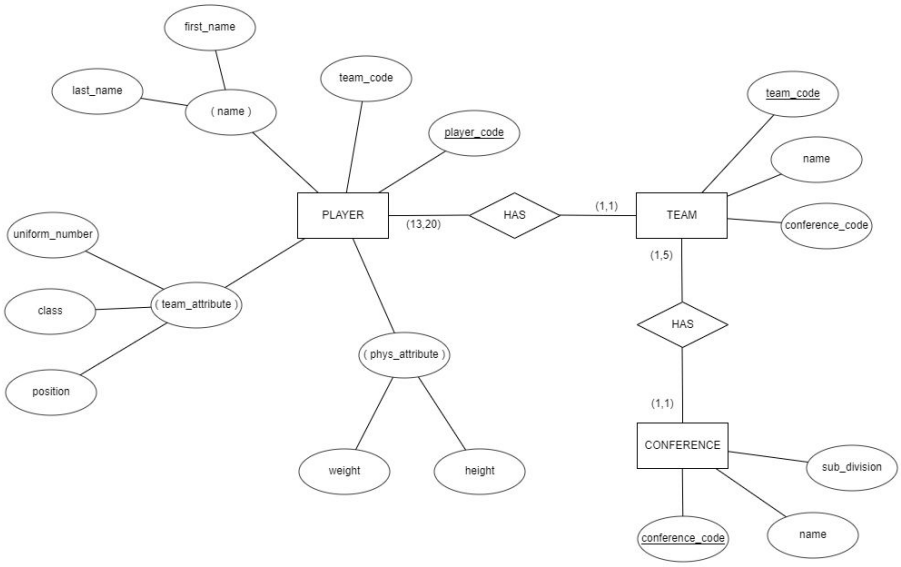
  ERROR 0: Scalar has more than one row in the output. 1st : (5,Akron,875), 2nd :(8,Alabama,911)

  Changing the '.' in 'team.team_code' to '::' would fix the error.

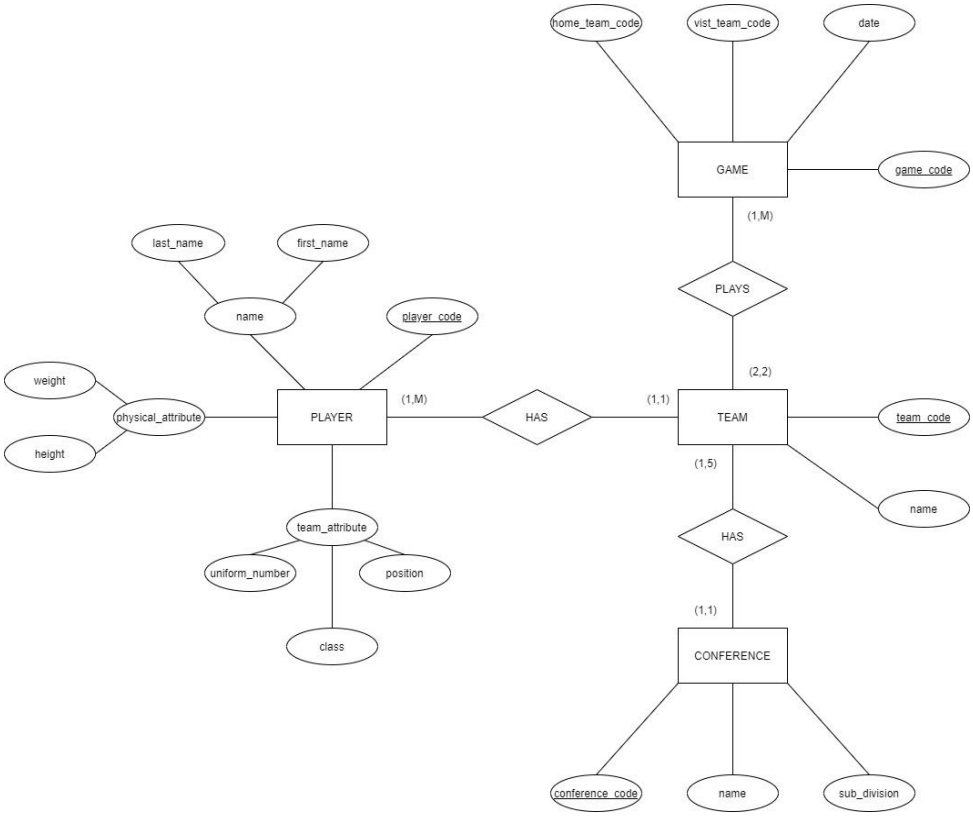
- For our fourth database, we were using a GAME_STATS dataset until we realized that it did not have a DATE attribute. A sample of the database we had before is as shown below:

Player Code	Game Code	Rush Yard	Rush TD	Pass Yard	Pass TD	Pass Int
85454	0005041920050917	153	2	79	2	1
85454	0129000520051001	0	1	285	2	0
85454	0005008620051008	-15	0	229	3	1
85454	0519000520051115	188	4	0	0	0
85454	0331000520051124	46	2	455	5	2
85454	0404000520051226	153	3	0	0	0
85455	0005041920050917	-2	0	132	1	0
85455	0503000520050924	20	0	4	0	0
85455	0725000520051022	0	0	15	0	0

We have decided to reserve this database for the later stages of the project as needed.



- The initial ER diagram was a rough draft to see what we needed to add, and it was including FOREIGN KEYs under every entity. This was revised into the diagram below.



- After meeting with the instructor, the diagram was revised into the version under ER Diagram to accurately portray the relationship between the TEAM and GAME entities.
- During transfer of files from Windows to Ubuntu we encountered issues that were resolved with either of two solutions. Error occurred saying fields could not load due to invalid data type. This is due to Windows file format lines terminating with \n and Unix format terminating with whitespace. Solution A: change control files to add "TERMINATED BY WHITESPACE" clause in the sqlldr control file. Solution B: install and run dos2unix linux application on data files prior to sqlldr call.

Previous Work

Description

This document presents the initial gathered requirements for both the hands-on design, and implementation of a database system. While the final result can certainly find use with application level main memory algorithms, the scope of this project's goals focus on information derived directly by database queries. The initial datasets detailed below will be expanded upon in the future by other related sets to add to the final derived query capabilities.

The project domain centers around archived United States college football (American) statistics. The sets listed below come from a common separated raw data provided by the Kaggle Data Science Company (http://www.kaggle.com/). Loaded data is currently limited to statistics based on 2005-2013 seasons, but our format should be easily compatible with both preceding and proceeding years.

Databases

PLAYER Table

Player Code	Team Code	Last Name	First Name	Uniform Number	Class	Position	Height	Weight
1054255	29	Wright	Scooby	31	FR	LB	73	230
1046216	9	Salako	Victor	77	FR	OL	78	315
1030872	8	Kouandjio	Arie	77	JR	OL	77	315
1054364	31	Roesler	Karl	96	FR	DL	75	235
1054238	28	Covey	Blake	86	SO	WR	73	204
1046285	29	Wood	Carter	66	SO	OL	74	272
1038713	51	Benenoch	Josh	37	SO	CB	70	200
1046181	8	Jones	Cyrus	5	SO	WR	70	196
1030940	28	Ozier	Kevin	82	SR	WR	74	200
1054212	9	Smith	Andra	99	FR	DL	76	305

The player.csv (2013) dataset contains 21794 records, or rows, of US college football players. A sample of 10 rows are shown in the picture. The dataset can be found at https://www.kaggle.com/mhixon/college-football-statistics/data? The following columns are included with the dataset:

Player Code column: The number that a player is assigned by the NCAA.

Team Code column: The team number that is assigned by the NCAA.

Last Name and First Name columns: The player's name.

Uniform Number column: The player's uniform number.

Class column: The player's academic year in 2013.

Position column: The player's position on the team.

Height column: contains the player's height in inches.

Weight column: contains the player's weight in pounds.

The team code attribute is a unique domain value that has a one:many join with the player code and is also used in the following dataset.

TEAM Table

Team Code	Name	Conference Code
5	Akron	875
8	Alabama	911
9	UAB	24312
28	Arizona State	905
29	Arizona	905
30	Arkansas State	818
31	Arkansas	911
37	Auburn	911
47	Ball State	875
51	Baylor	25354

The team.csv (2013) dataset contains 240 records of US college football teams. A sample of 10 rows are shown in the picture above. The dataset can also be found at https://www.kaggle.com/mhixon/college-football-statistics/data? The following columns are included with the dataset:

Team Code: A distinct integer unique to each team and an index that is referenced by other tables.

Name: An alphanumeric value designating the team name.

Conference Code: an integer referencing the conference code index in the conference dataset.

The team code attribute is a unique domain value that has a many:one join with the conference code in the following dataset.

CONFERENCE Table

Conference Code	Name	Subdivision
820	Atlantic 10	FCS
821	Atlantic Coast Conference	FBS
25354	Big 12 Conference	FBS
823	Big East Conference	FBS
825	Big Sky	FCS
826	Big South	FCS
827	Big Ten Conference	FBS
24312	Conference USA	FBS
853	Gateway Football Conference	FCS
99005	Ind	FCS

The conference.csv (2013) dataset contains 28 records of US college football conferences. A sample of 10 rows are shown in the picture above. The dataset can also be found at https://www.kaggle.com/mhixon/college-football-statistics/data? The following columns are included with the dataset:

Conference Code: a distinct integer unique to each conference and an index referenced by other tables

Name: an alpha numeric designating the conference's name

Subdivision: an alphanumeric with a limited input set designating the conference's subdivision

GAME Table

Game Code	Date	Visit Team Code	Home Team Code
706000000000000	11/2/2013	706	719
157000000000000	10/12/2013	157	28
697000000000000	10/12/2013	697	433
9040000000000	11/9/2013	9	388
454000000000000	8/31/2013	454	434
96100000000000	11/9/2013	96	811
277000000000000	11/9/2013	277	726
692000000000000	9/7/2013	692	796
774000000000000	10/5/2013	774	709
202000000000000	9/7/2013	202	367

The game.csv (2013) dataset contains 848 records of games played for US college football teams. A sample of 10 rows are shown in the picture above. The dataset can also be found at https://www.kaggle.com/mhixon/college-football-statistics/data? Obscure columns are detailed below:

Game Code: A unique key value to identify that particular game.

Application Specifications

This application is a US college football (American) fan resource for deriving statistical comparisons between players, teams and conferences. Some query examples include “compare height and weight ranges by position”, “change of player positions over time”, “graph visualization of school player consistency per conference”, or “compare player jersey duplication frequencies between teams” and “graph visualization of player transfers possibilities between teams”.

Relation Schemas

Primary keys were selected as unique and arbitrary numeric indices to also be used as foreign key references in their related entities. Information is expected to be complete and known prior to entry and so NULL values are not allowed. Although records in Team and Conference are indexed with team_code and conference_code respectively, each also has a name value which are expected to be distinct within each respective table.

TABLE PLAYER

player_code	NUMBER(7)	PRIMARY KEY
team_code	NUMBER(3)	FOREIGN KEY
last_name	VARCHAR(30)	NOT NULL
first_name	VARCHAR(30)	NOT NULL
uniform_number	NUMBER(2)	NOT NULL
class	CHAR(2)	NOT NULL
position	CHAR (2)	NOT NULL
height	NUMBER(2)	NOT NULL
weight	NUMBER(3)	NOT NULL

TABLE TEAM

team_code	NUMBER(3)	PRIMARY KEY
name	VARCHAR(30)	UNIQUE
conference_code	NUMBER(5)	FOREIGN KEY

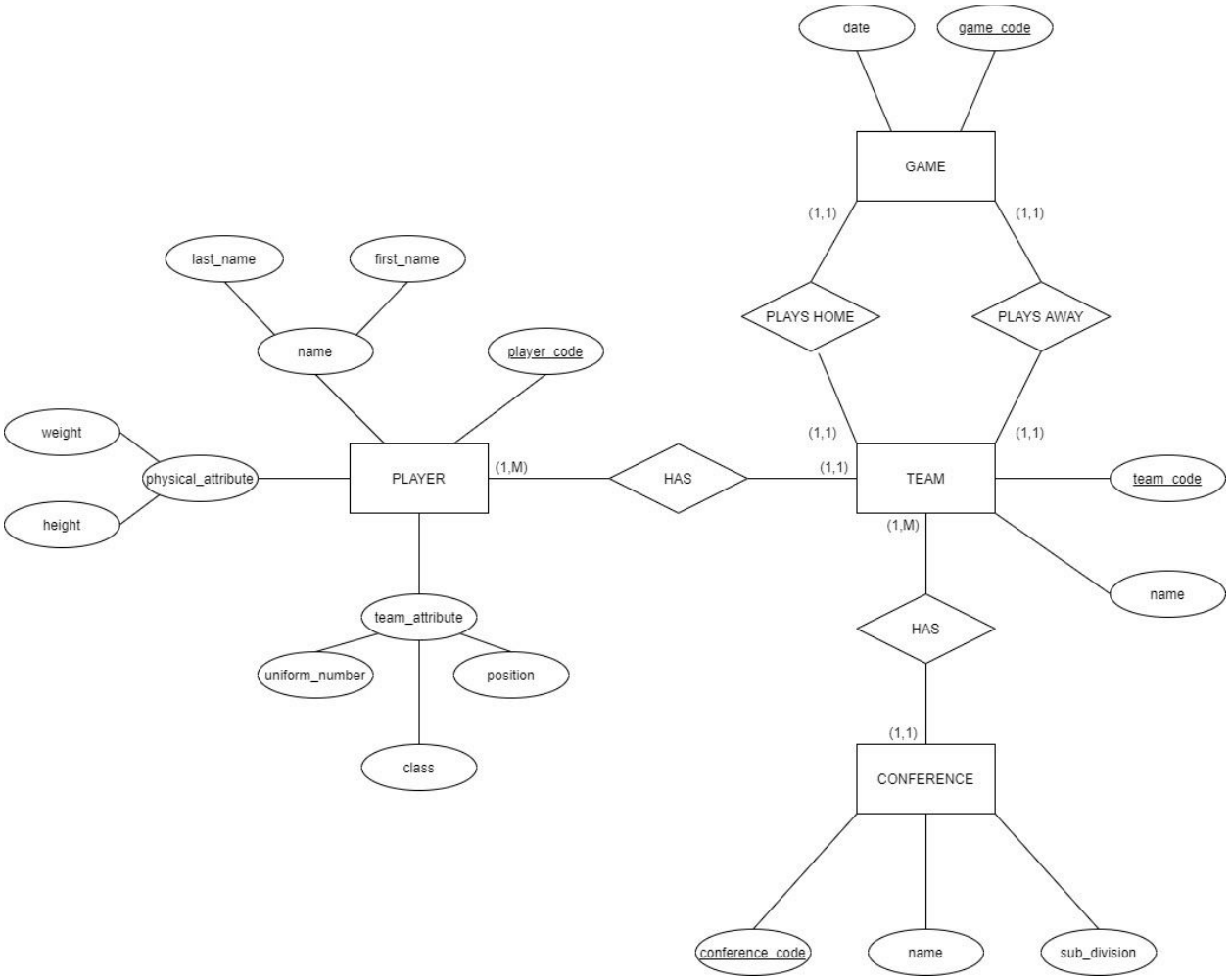
TABLE CONFERENCE

conference_code	NUMBER(5)	PRIMARY KEY
name	VARCHAR(30)	UNIQUE
sub_division	CHAR(3)	NOT NULL

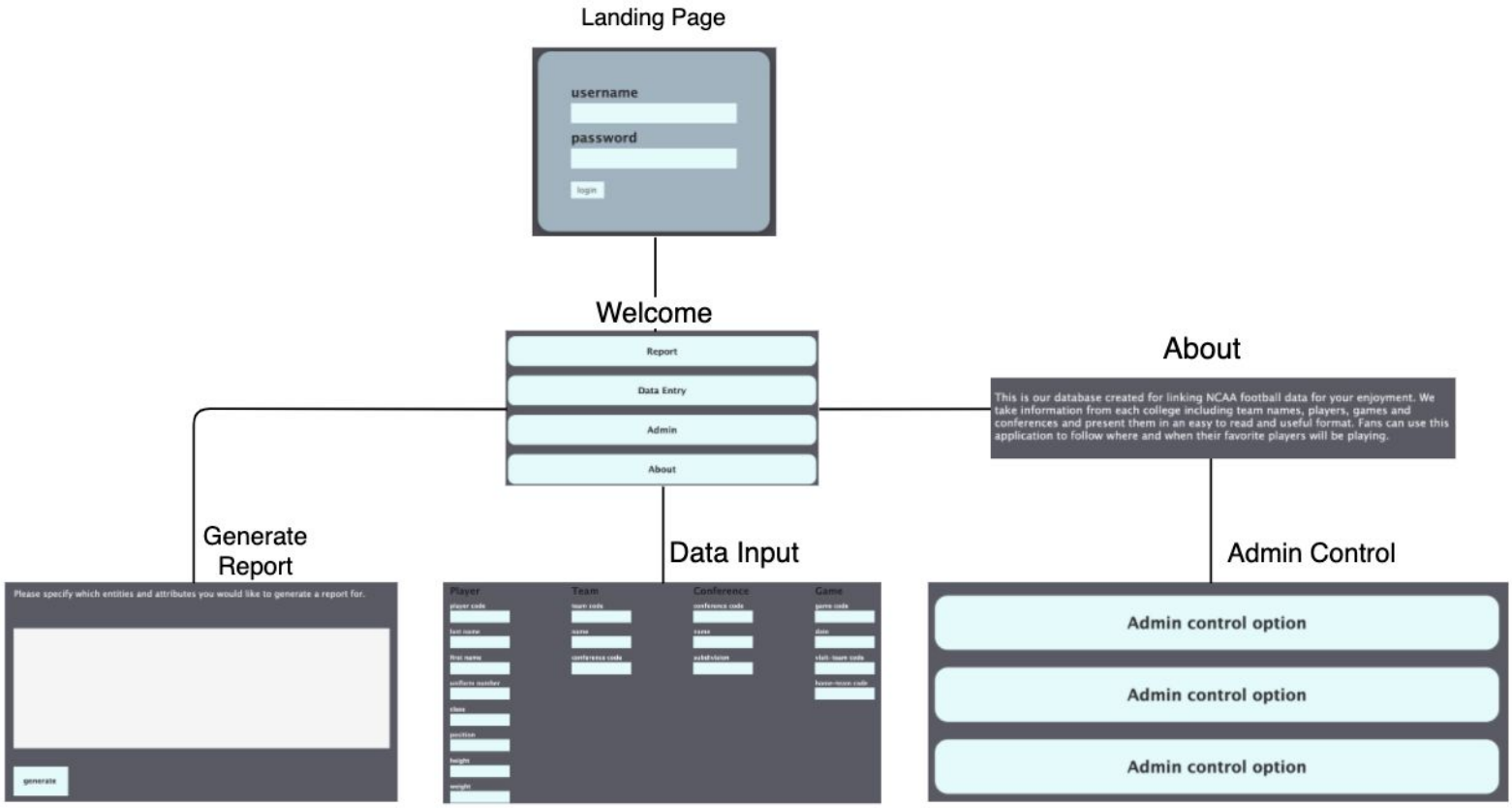
TABLE GAME

game_code	NUMBER(16)	PRIMARY KEY
date	DATE	NOT NULL
visit_team_code	NUMBER(3)	FOREIGN KEY
home_team_code	NUMBER(3)	FOREIGN KEY

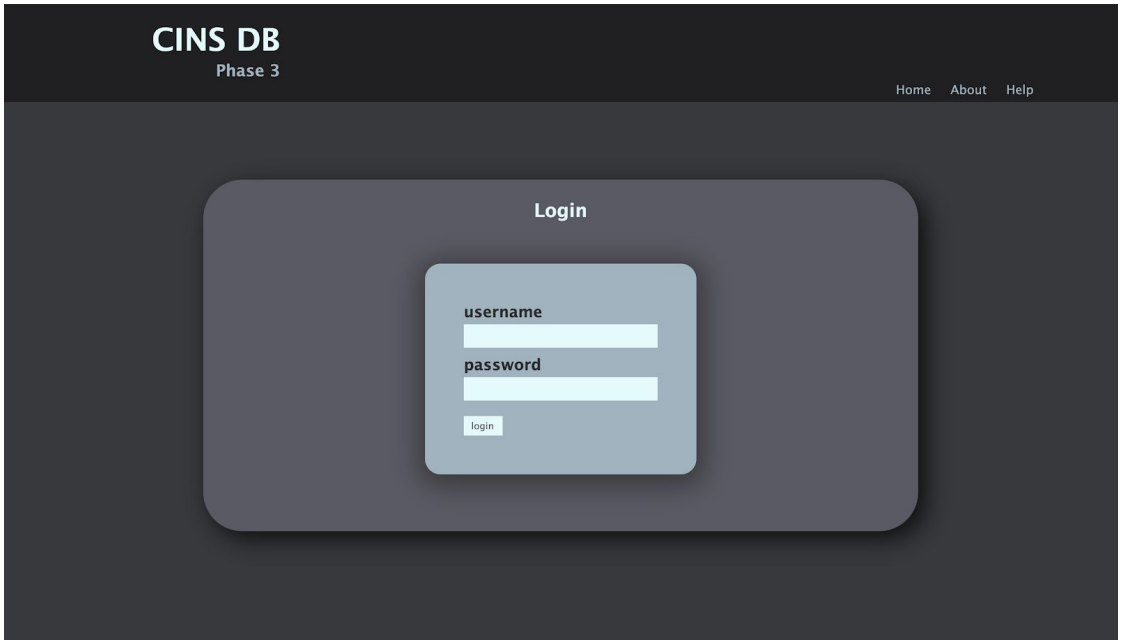
ER Diagram


date
game_code
GAME
(1,1)
(1,1)
PLAYS HOME
PLAYS AWAY
last_name
first_name
name
player_code
weight
physical_attribute
height
PLAYER
(1,M)
HAS
(1,1)
(1,1)
(1,1)
TEAM
team_code
(1,M)
name
team_attribute
uniform_number
position
class
HAS
(1,1)
CONFERENCE
conference_code
name
sub_division

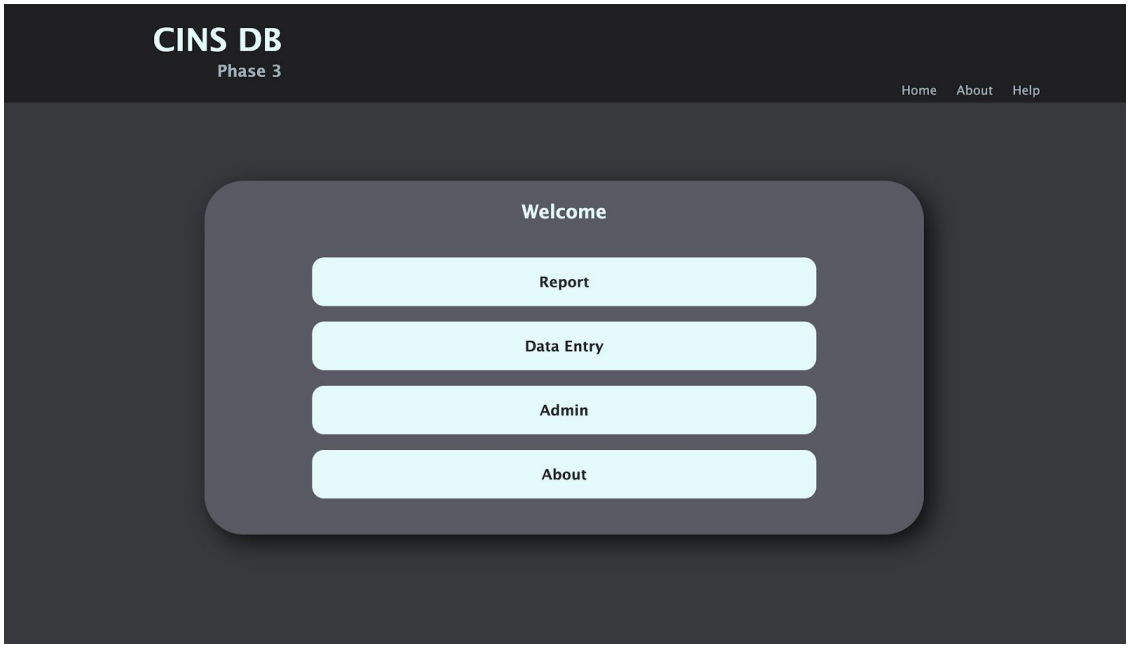
High Level Hierarchical Screen Layout Flow



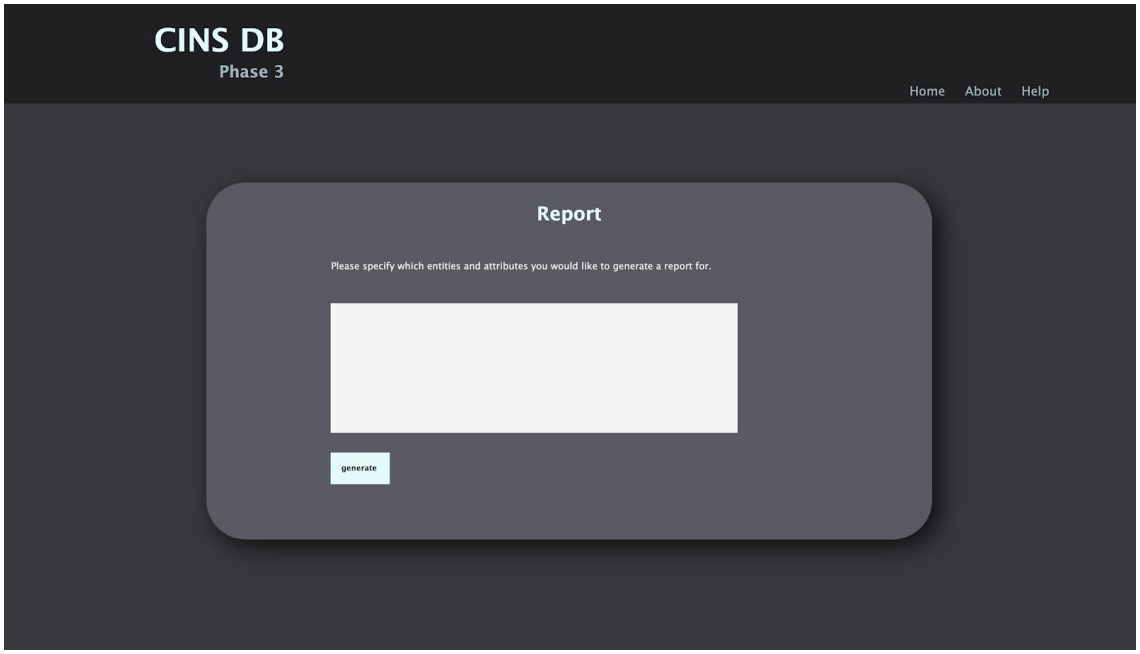
Login Page: The landing page presents the web user with a login screen to protect our database.



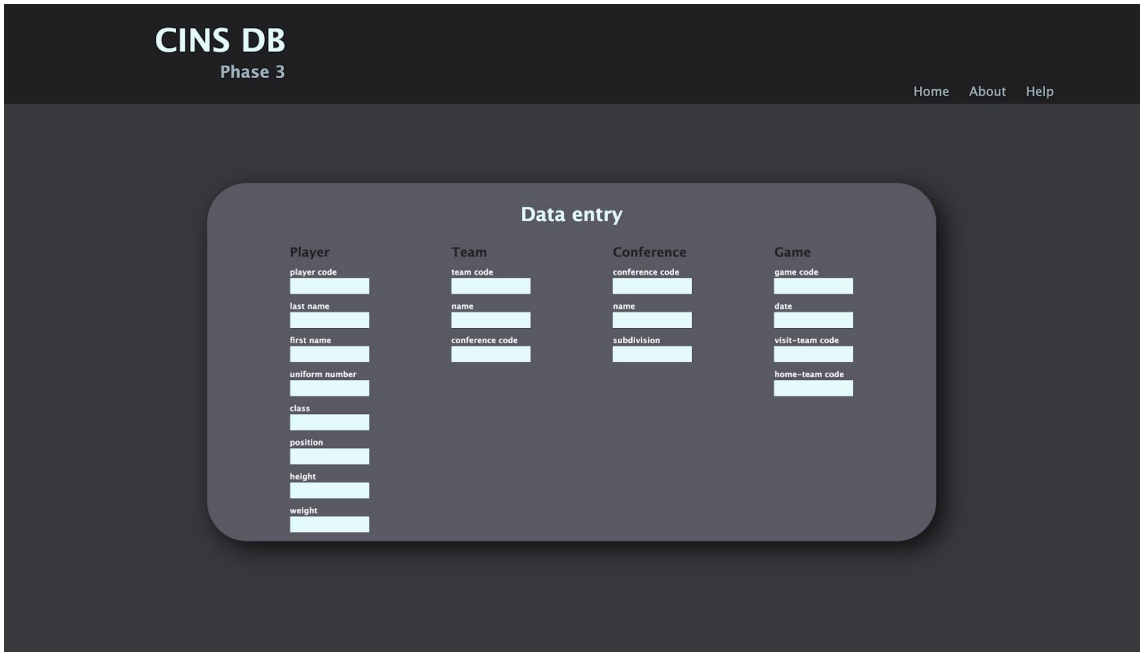
Welcome Page: The welcome page will provide options for the web user to choose their tasks.



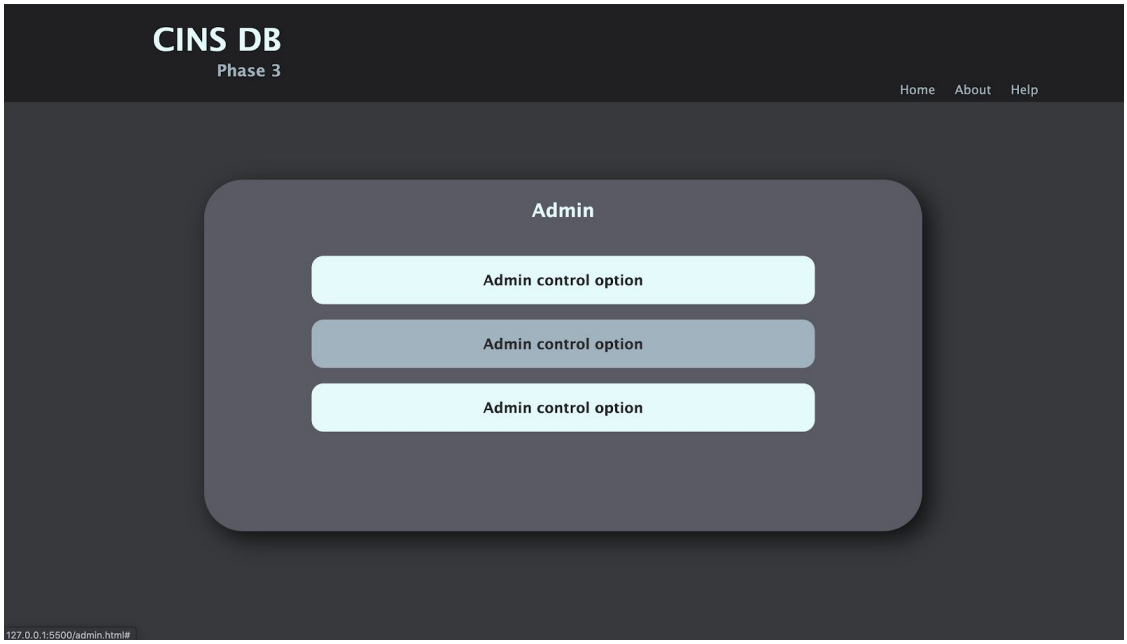
Generate report: On our generate report page, the users can easily work with the Oracle SQL database by using SQL syntax.



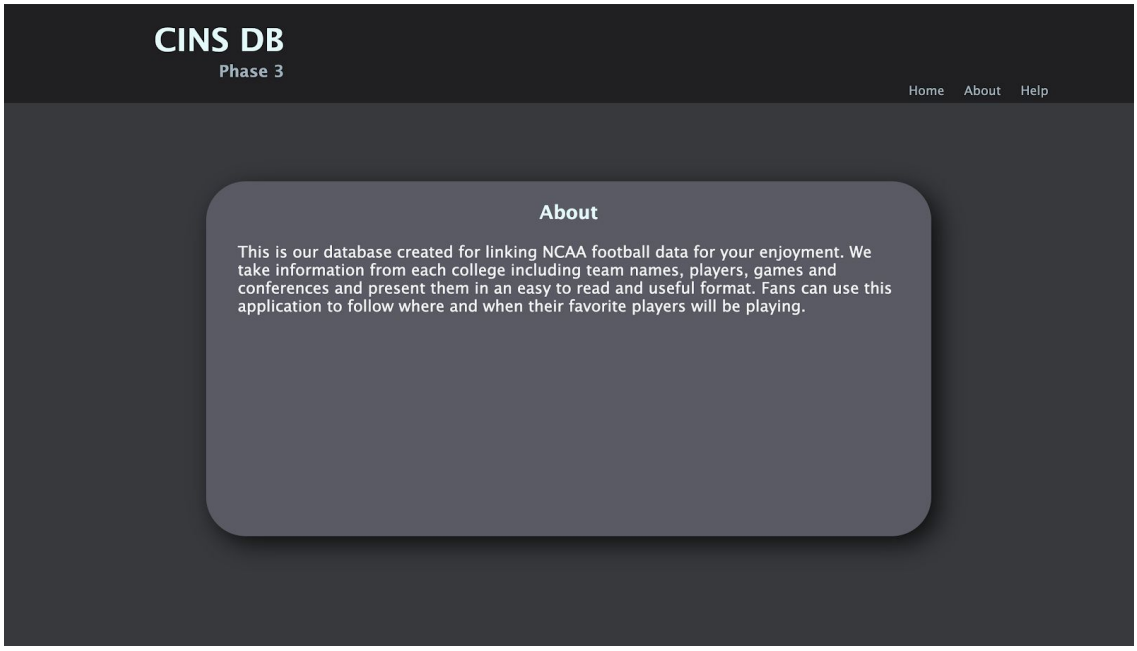
Data Entry: On our data entry page, users can easily add or update data to the database.



Admin Control: Our admin control gives access to various admin controls for our admins to work with the database.



About: A page that describes the topic of our project.



Load Data

- 4 Tables Loaded
    - Conference Rows Loaded 25
    - Team Rows Loaded 247
    - Player Rows Loaded 21794
    - Games Rows Loaded 848

Table Conference

CREATE TABLE Conference (conference_code NUMBER(5), conference_name VARCHAR2(30) UNIQUE,sub_division CHAR(3) NOT NULL, PRIMARY KEY (conference_code));

```
SQL> desc conference;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CONFERENCE_CODE                           NOT NULL NUMBER(5)
 CONFERENCE_NAME                                    VARCHAR2(30)
 SUB_DIVISION                              NOT NULL CHAR(3)

SQL>
```

Using sqlldr with load_conference.ctl:

```
ubuntu@ip-172-31-20-133: ~
ubuntu@ip-172-31-20-133:~$ sqlldr user101/pass101@xe control=load_conference.ctl

SQL*Loader: Release 11.2.0.2.0 - Production on Fri Jun 19 18:37:56 2020

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Commit point reached - logical record count 25
ubuntu@ip-172-31-20-133:~$
```

Cat of load_conference.log:

```
ubuntu@ip-172-31-20-133: ~

SQL*Loader: Release 11.2.0.2.0 - Production on Fri Jun 19 18:37:56 2020

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Control File:   load_conference.ctl
Data File:      2013conference.csv
  Bad File:     2013conference.bad
  Discard File:  none specified

 (Allow all discards)

Number to load: ALL
Number to skip: 0
Errors allowed: 50
Bind array:     64 rows, maximum of 256000 bytes
Continuation:    none specified
Path used:      Conventional

Table CONFERENCE, loaded from every logical record.
Insert option in effect for this table: INSERT

   Column Name                  Position   Len  Term Encl Datatype
------------------------------ ---------- ----- ---- ---- ---------------------
CONFERENCE_CODE                     FIRST     *   ,  O(") CHARACTER
CONFERENCE_NAME                      NEXT     *   ,  O(") CHARACTER
SUB_DIVISION                         NEXT     *   ,  O(") CHARACTER


Table CONFERENCE:
  25 Rows successfully loaded.
  0 Rows not loaded due to data errors.
  0 Rows not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.


Space allocated for bind array:                  49536 bytes(64 rows)
Read  buffer bytes: 1048576

Total logical records skipped:          0
Total logical records read:            25
Total logical records rejected:         0
Total logical records discarded:        0

Run began on Fri Jun 19 18:37:56 2020
Run ended on Fri Jun 19 18:37:57 2020

Elapsed time was:     00:00:00.16
CPU time was:         00:00:00.01
ubuntu@ip-172-31-20-133:~$
```

First 10 records:

```
ubuntu@ip-172-31-20-133: ~
SQL> select * from conference where rownum <= 10;

CONFERENCE_CODE CONFERENCE_NAME                SUB
--------------- ------------------------------ ---
            823 American Athletic Conference   FBS
            821 Atlantic Coast Conference      FBS
          25354 Big 12 Conference              FBS
            827 Big Ten Conference             FBS
          24312 Conference USA                 FBS
          99001 Independent                    FBS
            875 Mid-American Conference        FBS
           5486 Mountain West Conference       FBS
            905 Pac-12 Conference              FBS
            911 Southeastern Conference        FBS

10 rows selected.

SQL>
```

Table Team

CREATE TABLE Team (team_code NUMBER, team_name VARCHAR2(30) UNIQUE, conference_code NUMBER NOT NULL, PRIMARY KEY (team_code), FOREIGN KEY (conference_code) REFERENCES Conference(conference_code));

```
SQL> CREATE TABLE Team (team_code NUMBER, team_name VARCHAR2(30) UNIQUE, conference_code NUMBER NOT NULL, PRIMARY KEY (t
eam_code), FOREIGN KEY (conference_code) REFERENCES Conference(conference_code));

Table created.

SQL> desc team;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TEAM_CODE                                 NOT NULL NUMBER
 TEAM_NAME                                          VARCHAR2(30)
 CONFERENCE_CODE                           NOT NULL NUMBER

SQL>
```

Using sqlldr with load_team.ctl:

```
ubuntu@ip-172-31-20-133:~$ sqlldr user101/pass101@xe control=load_team.ctl

SQL*Loader: Release 11.2.0.2.0 - Production on Fri Jun 19 20:03:21 2020

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Commit point reached - logical record count 64
Commit point reached - logical record count 128
Commit point reached - logical record count 192
Commit point reached - logical record count 247
ubuntu@ip-172-31-20-133:~$
```

Cat of load_team.log:

```
SQL*Loader: Release 11.2.0.2.0 - Production on Fri Jun 19 20:03:21 2020

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Control File:   load_team.ctl
Data File:      2013team.csv
  Bad File:     2013team.bad
  Discard File:  none specified

 (Allow all discards)

Number to load: ALL
Number to skip: 0
Errors allowed: 50
Bind array:     64 rows, maximum of 256000 bytes
Continuation:    none specified
Path used:      Conventional

Table TEAM, loaded from every logical record.
Insert option in effect for this table: INSERT

   Column Name                  Position   Len  Term Encl Datatype
------------------------------ ---------- ----- ---- ---- ---------------------
TEAM_CODE                           FIRST     *   ,  O(") CHARACTER
TEAM_NAME                            NEXT     *   ,  O(") CHARACTER
CONFERENCE_CODE                      NEXT     8           INTEGER


Table TEAM:
  247 Rows successfully loaded.
  0 Rows not loaded due to data errors.
  0 Rows not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.


Space allocated for bind array:                  33536 bytes(64 rows)
Read  buffer bytes: 1048576

Total logical records skipped:          0
Total logical records read:           247
Total logical records rejected:         0
Total logical records discarded:        0

Run began on Fri Jun 19 20:03:21 2020
Run ended on Fri Jun 19 20:03:21 2020

Elapsed time was:     00:00:00.05
CPU time was:         00:00:00.00
ubuntu@ip-172-31-20-133:~$
```

First 10 records in table:

```
SQL> select * from team where rownum <= 10;

 TEAM_CODE TEAM_NAME                      CONFERENCE_CODE
---------- ------------------------------ ---------------
         5 Akron                                     875
         8 Alabama                                   911
         9 UAB                                     24312
        28 Arizona State                             905
        29 Arizona                                   905
        30 Arkansas State                            818
        31 Arkansas                                  911
        37 Auburn                                    911
        47 Ball State                                875
        51 Baylor                                  25354

10 rows selected.

SQL>
```

Table Player

CREATE TABLE Player (player_code NUMBER, team_code NUMBER NOT NULL, last_name VARCHAR2(30) NOT NULL, first_name VARCHAR2(30) NOT NULL, uniform_number VARCHAR(3), class CHAR(2), position VARCHAR2(5), height NUMBER, weight NUMBER, PRIMARY KEY (player_code), FOREIGN KEY (team_code) REFERENCES Team(team_code));

```
SQL> CREATE TABLE Player (player_code NUMBER, team_code NUMBER NOT NULL, last_name VARCHAR2(30) NOT NULL, first_name
 VARCHAR2(30) NOT NULL, uniform_number VARCHAR(3), class CHAR(2), position VARCHAR2(5), height NUMBER, weight NUMBER
, PRIMARY KEY (player_code), FOREIGN KEY (team_code) REFERENCES Team(team_code));

Table created.

SQL> desc player;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PLAYER_CODE                               NOT NULL NUMBER
 TEAM_CODE                                 NOT NULL NUMBER
 LAST_NAME                                 NOT NULL VARCHAR2(30)
 FIRST_NAME                                NOT NULL VARCHAR2(30)
 UNIFORM_NUMBER                                     VARCHAR2(3)
 CLASS                                              CHAR(2)
 POSITION                                           VARCHAR2(5)
 HEIGHT                                             NUMBER
 WEIGHT                                             NUMBER

SQL>
```

Using sqlldr with load_player.ctl:

```
ubuntu@ip-172-31-20-133:~$ sqlldr user101/pass101@xe control=load_player.

SQL*Loader: Release 11.2.0.2.0 - Production on Fri Jun 19 21:05:01 2020

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reser

Commit point reached - logical record count 110
Commit point reached - logical record count 220
Commit point reached - logical record count 330
Commit point reached - logical record count 440
Commit point reached - logical record count 550
Commit point reached - logical record count 660
Commit point reached - logical record count 770
Commit point reached - logical record count 880
Commit point reached - logical record count 990
Commit point reached - logical record count 1100
```

……..

```
Commit point reached - logical record count 20900
Commit point reached - logical record count 21010
Commit point reached - logical record count 21120
Commit point reached - logical record count 21230
Commit point reached - logical record count 21340
Commit point reached - logical record count 21450
Commit point reached - logical record count 21560
Commit point reached - logical record count 21670
Commit point reached - logical record count 21780
Commit point reached - logical record count 21794
ubuntu@ip-172-31-20-133:~$
```

Cat of load_player.log:

```
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Control File:   load_player.ctl
Data File:      2013player.csv
  Bad File:     2013player.bad
  Discard File:  none specified

 (Allow all discards)

Number to load: ALL
Number to skip: 0
Errors allowed: 50
Bind array:     64 rows, maximum of 256000 bytes
Continuation:    none specified
Path used:      Conventional

Table PLAYER, loaded from every logical record.
Insert option in effect for this table: INSERT
TRAILING NULLCOLS option in effect

   Column Name                  Position   Len  Term Encl Datatype
------------------------------ ---------- ----- ---- ---- ---------------------
PLAYER_CODE                         FIRST     *   ,  O(") CHARACTER
TEAM_CODE                            NEXT     *   ,  O(") CHARACTER
LAST_NAME                            NEXT     *   ,  O(") CHARACTER
FIRST_NAME                           NEXT     *   ,  O(") CHARACTER
UNIFORM_NUMBER                       NEXT     *   ,  O(") CHARACTER
CLASS                                NEXT     *   ,  O(") CHARACTER
POSITION                             NEXT     *   ,  O(") CHARACTER
HEIGHT                               NEXT     *   ,  O(") CHARACTER
WEIGHT                               NEXT     *   ,  O(") CHARACTER


Table PLAYER:
  21794 Rows successfully loaded.
  0 Rows not loaded due to data errors.
  0 Rows not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.


Space allocated for bind array:                 148608 bytes(64 rows)
Read   buffer bytes: 1048576

Total logical records skipped:          0
Total logical records read:         21794
Total logical records rejected:         0
Total logical records discarded:        0

Run began on Fri Jun 19 21:13:07 2020
Run ended on Fri Jun 19 21:13:08 2020

Elapsed time was:     00:00:01.20
CPU time was:         00:00:00.08
ubuntu@ip-172-31-20-133:~$
```

First 10 records in table:

```
SQL> select * from player where rownum <= 10;

PLAYER_CODE  TEAM_CODE LAST_NAME                      FIRST_NAME                     UNI CL POSIT     HEIGHT     WEIGHT
----------- ---------- ------------------------------ ------------------------------ --- -- ----- ---------- ----------
    1022870          5 Alexander                      Bill                            21  SR CB            68        172
    1022862          5 Alexander                      Broderick                       2   SR FB            72        225
    1046129          5 Allen                          Christian                       49  FR FB            72        249
    1035204          5 Allen                          Jeff                            46  JR DL            74        262
    1030820          5 Bailey                         Austin                          40  SR DL            71        262
    1054125          5 Berger                         Brady                           36  FR WR            73        197
    1053898          5 Bice                           Nick                            14  JR WR            68        153
    1054121          5 Bickley                        Fransohn                        24  FR WR            66        138
    1038393          5 Bohan                          Andrew                          85  SO OL            75        280
    1054138          5 Brittnum                       Cedric                          74  JR OL            76        316

10 rows selected.

SQL>
```

Table Games

CREATE TABLE Game (game_code NUMBER, game_date DATE NOT NULL, visit_team_code NUMBER, home_team_code NUMBER, PRIMARY KEY (game_code), FOREIGN KEY (visit_team_code) REFERENCES Team(team_code), FOREIGN KEY (home_team_code) REFERENCES Team(team_code));

```
SQL> CREATE TABLE Game (game_code NUMBER, game_date DATE NOT NULL, visit_team_code NUMBER, home_team_code NUMBER, PR
IMARY KEY (game_code), FOREIGN KEY (visit_team_code) REFERENCES Team(team_code), FOREIGN KEY (home_team_code) REFERE
NCES Team(team_code));

Table created.

SQL> desc game;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 GAME_CODE                                 NOT NULL NUMBER
 GAME_DATE                                 NOT NULL DATE
 VISIT_TEAM_CODE                                    NUMBER
 HOME_TEAM_CODE                                     NUMBER

SQL>
```

Using sqlldr with load_game.ctl:

```
ubuntu@ip-172-31-20-133:~$ ubuntu@ip-172-31-20-133:~$ sqlldr user101/pass101@xe control=load_game.ctl

SQL*Loader: Release 11.2.0.2.0 - Production on Fri Jun 19 21:51:40 2020

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Commit point reached - logical record count 64
Commit point reached - logical record count 128
Commit point reached - logical record count 192
Commit point reached - logical record count 256
Commit point reached - logical record count 320
Commit point reached - logical record count 384
Commit point reached - logical record count 448
Commit point reached - logical record count 512
Commit point reached - logical record count 576
Commit point reached - logical record count 640
Commit point reached - logical record count 704
Commit point reached - logical record count 768
Commit point reached - logical record count 832
Commit point reached - logical record count 848
ubuntu@ip-172-31-20-133:~$
```

Cat of load_game.log:

```
ubuntu@ip-172-31-20-133:~$ cat load_game.log

SQL*Loader: Release 11.2.0.2.0 - Production on Fri Jun 19 21:51:40 2020

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

Control File:   load_game.ctl
Data File:      2013game.csv
  Bad File:     2013game.bad
  Discard File:  none specified

 (Allow all discards)

Number to load: ALL
Number to skip: 0
Errors allowed: 50
Bind array:     64 rows, maximum of 256000 bytes
Continuation:    none specified
Path used:      Conventional

Table GAME, loaded from every logical record.
Insert option in effect for this table: INSERT

   Column Name                  Position   Len  Term Encl Datatype
------------------------------ ---------- ----- ---- ---- ---------------------
GAME_CODE                           FIRST     *   ,  O(") CHARACTER
GAME_DATE                            NEXT     *   ,  O(") DATE MM/DD/YYYY
VISIT_TEAM_CODE                      NEXT     *   ,  O(") CHARACTER
HOME_TEAM_CODE                       NEXT     *   ,  O(") CHARACTER


Table GAME:
  848 Rows successfully loaded.
  0 Rows not loaded due to data errors.
  0 Rows not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.


Space allocated for bind array:                  66048 bytes(64 rows)
Read  buffer bytes: 1048576

Total logical records skipped:          0
Total logical records read:           848
Total logical records rejected:         0
Total logical records discarded:        0

Run began on Fri Jun 19 21:51:40 2020
Run ended on Fri Jun 19 21:51:40 2020

Elapsed time was:     00:00:00.10
CPU time was:         00:00:00.01
ubuntu@ip-172-31-20-133:~$
```

First 10 records in table:

```
SQL> select to_char(game_code) as gamecode, game_date, visit_team_code, home_team_code from game where rownum <= 10;

GAMECODE                                GAME_DATE VISIT_TEAM_CODE HOME_TEAM_CODE
--------------------------------------- --------- --------------- --------------
5012820130829                           29-AUG-13               5            128
102063020130829                         29-AUG-13             102            630
299004720130829                         29-AUG-13             299             47
305030620130829                         29-AUG-13             305            306
314071820130829                         29-AUG-13             314            718
355033120130829                         29-AUG-13             355            331
433073620130829                         29-AUG-13             433            736
457064820130829                         29-AUG-13             457            648
465042820130829                         29-AUG-13             465            428
587009620130829                         29-AUG-13             587             96

10 rows selected.

SQL>
```

SQL Command Examples

SELECT and UPDATE example:

```
SQL> desc conference;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CONFERENCE_CODE                           NOT NULL NUMBER(5)
 CONFERENCE_NAME                                    VARCHAR2(30)
 SUB_DIVISION                              NOT NULL CHAR(3)

SQL> select *from conference;

CONFERENCE_CODE CONFERENCE_NAME                SUB
--------------- ------------------------------ ---
            823 American Athletic Conference   FBS
            821 Atlantic Coast Conference      FBS
          25354 Big 12 Conference              FBS
            827 Big Ten Conference             FBS
          24312 Conference USA                 FBS
          99001 Independent                    FBS
            875 Mid-American Conference        FBS
           5486 Mountain West Conference       FBS
            905 Pac-12 Conference              FBS
            911 Southeastern Conference        FBS
            818 Sun Belt Conference            FBS
            825 Big Sky                        FCS
            826 Big South                      FCS
            837 Colonial                       FCS
          99005 Ind                            FCS
            865 Ivy                            FCS
            876 Mid-Eastern                    FCS
            853 MVFC                           FCS
            846 Northeast                      FCS
            902 OVC                            FCS
            838 Patriot                        FCS
          21451 Pioneer                        FCS
            912 Southern                       FCS
            914 Southland                      FCS
            916 Southwestern                   FCS

25 rows selected.

SQL> UPDATE CONFERENCE
  2  SET conference_name = 'TEST TEST TEST'
  3  WHERE conference_code = 912;

1 row updated.

SQL> select *from conference;

CONFERENCE_CODE CONFERENCE_NAME                SUB
--------------- ------------------------------ ---
            823 American Athletic Conference   FBS
            821 Atlantic Coast Conference      FBS
          25354 Big 12 Conference              FBS
            827 Big Ten Conference             FBS
          24312 Conference USA                 FBS
          99001 Independent                    FBS
            875 Mid-American Conference        FBS
           5486 Mountain West Conference       FBS
            905 Pac-12 Conference              FBS
            911 Southeastern Conference        FBS
            818 Sun Belt Conference            FBS
            825 Big Sky                        FCS
            826 Big South                      FCS
            837 Colonial                       FCS
          99005 Ind                            FCS
            865 Ivy                            FCS
            876 Mid-Eastern                    FCS
            853 MVFC                           FCS
            846 Northeast                      FCS
            902 OVC                            FCS
            838 Patriot                        FCS
          21451 Pioneer                        FCS
            912 TEST TEST TEST                 FCS
            914 Southland                      FCS
            916 Southwestern                   FCS

25 rows selected.
```

'Southern' in the CONFERENCE table was updated to 'TEST TEST TEST'

JOIN example:

```
SQL> SELECT * FROM TEAM WHERE rownum <=10;

 TEAM_CODE TEAM_NAME                      CONFERENCE_CODE
---------
         5 Akron                                    875
         8 Alabama                                  911
         9 UAB                                    24312
        28 Arizona State                            905
        29 Arizona                                  905
        30 Arkansas State                           818
        31 Arkansas                                 911
        37 Auburn                                   911
        47 Ball State                               875
        51 Baylor                                 25354

10 rows selected.

SQL> SELECT * FROM PLAYER WHERE team_code = 5 AND rownum <=4;

PLAYER_CODE  TEAM_CODE LAST_NAME                      FIRST_NAME                     UNI CL POSIT     HEIGHT     WEIGHT
----------- ---------- ------------------------------ ------------------------------ --- -- ----- ---------- ----------
    1022870          5 Alexander                      Bill                            21  SR CB           68        172
    1022862          5 Alexander                      Broderick                       2   SR FB           72        225
    1046129          5 Allen                          Christian                       49  FR FB           72        249
    1035204          5 Allen                          Jeff                            46  JR DL           74        262

SQL> SELECT PLAYER.player_code, TEAM.team_name, PLAYER.last_name, PLAYER.first_name, PLAYER.uniform_number, PLAYER.class, PLAYER.position, PLAYER.height, PLAYER.weight
  2  FROM PLAYER
  3  INNER JOIN TEAM
  4  ON PLAYER.team_code = TEAM.team_code
  5  WHERE rownum <=5;

PLAYER_CODE TEAM_NAME                      LAST_NAME                      FIRST_NAME                     UNI CL POSIT    HEIGHT     WEIGHT
----------- ------------------------------ ------------------------------ ------------------------------ --- -- ----- ---------- ----------
    1022870 Akron                          Alexander                      Bill                            21  SR CB           68        172
    1022862 Akron                          Alexander                      Broderick                       2   SR FB           72        225
    1046129 Akron                          Allen                          Christian                       49  FR FB           72        249
    1035204 Akron                          Allen                          Jeff                            46  JR DL           74        262
    1030820 Akron                          Bailey                         Austin                          40  SR DL           71        262

SQL>
```

The PLAYER table merged with the TEAM table to display the team names instead of just thier team codes

CREATE VIEW example:

```
SQL> CREATE VIEW ARIZONA_STATE
  2  AS SELECT first_name,last_name
  3  FROM PLAYER
  4  WHERE TEAM_CODE = 28;

View created.

SQL> SELECT first_name, last_name
  2  FROM ARIZONA_STATE
  3  WHERE rownum <=20;

FIRST_NAME                     LAST_NAME
------------------------------ ------------------------------
Alonzo                         Agwuenu
Sil                            Ajawara
Dante                          Alexander
Marcus                         Ball
Mike                           Bercovici
Ezekiel                        Bishop
Carl                           Bradford
Dwain                          Bradshaw
Alex                           Bykovskiy
Lloyd                          Carrington
Gary                           Chambers
Demetrius                      Cherry
Taylor                         Cohan
Davon                          Coleman
Gannon                         Conway
Blake                          Covey
Chans                          Cox
Chris                          Coyle
Alden                          Darby
Billy                          Davis

20 rows selected.
```

Displaying 20 players from Arizona State using CREATE VIEW.

GROUP BY example:

```
SQL> set pagesize 1000
SQL> SELECT PLAYER.team_code, TEAM.team_name, COUNT (*)
  2  FROM TEAM,PLAYER
  3  WHERE PLAYER.team_code = TEAM.team_code
  4  GROUP BY PLAYER.team_code, TEAM.team_name
  5  ORDER BY COUNT(*) ASC;

 TEAM_CODE TEAM_NAME                        COUNT(*)
---------- ------------------------------ ----------
       676 Stephen F. Austin                      81
        17 Alcorn St.                             82
       667 Southern Utah                          82
       261 Grambling                              82
       228 Florida A&M                            82
       314 Jackson State                          83
         6 Alabama A&M                            84
       153 Colgate                                84
       294 Idaho State                            84
      2678 Arkansas-Pine Bluff                    84
       236 Fordham                                85
       647 South Carolina State                   85
      1092 Gardner-Webb                           86
       498 Louisiana-Monroe                       87
       244 Furman                                 87
       632 Savannah State                         87
       758 Weber State                            87
       693 Chattanooga                            88
       290 Howard                                 89
       454 Murray State                           89
       575 Richmond                               89
       201 Eastern Illinois                       89
       711 Towson                                 89
       650 South Dakota                           89
        61 Bethune-Cookman                        90
```

```
       235 Florida                                120
       365 LSU                                    121
       433 Mississippi                            122
       428 Minnesota                              122
       311 Iowa State                             124
       107 California                             124
       253 Georgia Southern                       124
         2 Abilene Christian                      124
       312 Iowa                                   126
       257 Georgia                                129
       521 Oklahoma State                         130
       434 Missouri                               134
       463 Nebraska                               136
       327 Kansas State                           137
       725 Army                                   150
       726 Navy                                   161

207 rows selected.
```

Grouping the teams by the amount of players they have or had. Not all 207 rows are shown.