Code

Project / Design & Implement a Relational Database

/* "Below is the breakdown the tasks: • The first installing package (RMySQL) in the library library • Then, setting AWS-database • Dropped the tables • Created table and constraints: o Created table Incidents, o Created table airports, o Altered table incidents adding constrain Foreign Key origin references to airport(iaid) o Created table conditions o Altered table incidents adding constrain foreign key incidents condition FK references condition (cid) o Altered table incidents add constrain flight phases in such: takeoff, landing, inflight, unknown • load data from CSV file: o Change the colname of one column o insert to conditions table o insert to airports o retrieve from airports o insert into incidents o Do not select MILITARY: choose the foreign key for airport, choose the foreign key for sky_conditions o rename columns o incidents insert to incidents table • Query selecting airlines, count from incident grouping descent and limiting to 10 airlines. • Query selecting flight phase counting & selecting from incident grouping by flight phase, and calculating the average number of bird strike incidents during any flight phase. • Query selecting months (date) , count from incidents group by month. • Query buildings a line char visual, scatter plot (select count bird strikes, year between 2005-2011. X-axis = year, and Y-axis birdstrikes incidents • Created a store procedure in MySQL, calling insert incidents, and finally sql chunk selecting from incidents where rid = '200099' • Lastly, disconnect or close the database.

Challenges: The directions were very helpful, but I experienced some challenges with the database setup and running. I created db#tree twice both times failed upon loading (sign: loading local data is disabled; thus must be enabled on both the client and server sides). Then, I create Amazon AWS. Afterward, our program was able to connect and run without any issues."*/

Hide

```
# 1. Library
library(RMySQL)


# 2. Settings

# AWS
db_user <- 'admin'
db_password <- '####'
db_name <- 'cs_Database_Project1'
db_host <- 'mysqlworks.amazonaws.com'
db_port <- 2211

# 3. Read data from db
mydb <-  dbConnect(MySQL(), user = db_user, password = db_password,
                   dbname = db_name, host = db_host, port = db_port)
```

#-Drop Table-

Hide

```
DROP TABLE IF EXISTS incidents
```

Hide

```
DROP TABLE IF EXISTS airports
```

Hide

```
DROP TABLE IF EXISTS conditions
```

Hide

```
DROP PROCEDURE IF EXISTS  InsertIncidents
```

#create table and constrains

Hide

```
CREATE TABLE incidents(
    rid INT PRIMARY KEY,
    date DATE,
    origin INT,
    airline VARCHAR(50) DEFAULT 'unknown',
    aircraft VARCHAR(50),
    flightPhase VARCHAR(50),
    altitude VARCHAR(50),
    conditions INT,
    warning BOOLEAN

)
```

Hide

```
CREATE TABLE airports(
  aid int PRIMARY KEY AUTO_INCREMENT,
  airportName VARCHAR (150) unique DEFAULT 'unknown',
  airportCode VARCHAR(50),
  state VARCHAR(50)

)
```

Hide

```
ALTER TABLE incidents
add constraint fk_incidents_origin
FOREIGN KEY (origin) REFERENCES airports(aid);
```

Hide

```
CREATE TABLE conditions(

  cid INT PRIMARY KEY AUTO_INCREMENT,
  conditions VARCHAR (100),
  explanation VARCHAR(256)
 )
```

Hide

```
ALTER TABLE incidents
ADD constraint fk_incidents_conditions
 FOREIGN KEY (conditions) REFERENCES conditions(cid);
```

Hide

```
ALTER TABLE incidents
ADD constraint chk_incidents_phase
check(flightPhase IN ('takeoff',
 'landing', 'inflight', 'unknown'));
```

#load data from csv file

Hide

```
birdStrikesData <- read.csv(file="BirdStrikesData-V2.csv", header=TRUE, sep=",")
sky_conditions <- unique(birdStrikesData['sky_conditions'])
```

Change colname of one column

Hide

```
colnames(sky_conditions)[colnames(sky_conditions) == "sky_conditions"] <- "conditions"
```

insert to conditions table

Hide

```
dbWriteTable(mydb, "conditions", sky_conditions, overrite=FALSE, append=TRUE, row.names=FALSE)
```

insert to airports

Hide

```
airports <- unique(birdStrikesData[c('airport', 'origin')])
colnames(airports)[colnames(airports) == "airport"] <- "airportName"
colnames(airports)[colnames(airports) == "origin"] <- "state"
dbWriteTable(mydb, 'airports', airports, override=FALSE, append=TRUE, row.names=FALSE)
```

#retrieve from airports

Hide

```
rs <- dbSendQuery(mydb, "SELECT aid, airportName FROM airports")
aid_airport_names <-  fetch(rs, n = -1)
```

Hide

```
rs <- dbSendQuery(mydb, "SELECT cid, conditions FROM conditions;")
cid_conditions <-  fetch(rs, n = -1)
```

#insert into incidents

Hide

```
incidents = birdStrikesData[c('rid', 'flight_date', 'airport', 'aircraft', 'airline', 'flight_phase', 'altitude_ft', 'sky_conditions', 'pilot_warned_flag')]
```

Do not select MILITARY

Hide

```
incidents = incidents[incidents$airline != 'MILITARY',]
for(i in 1:nrow(incidents)) {

  #choose the foreign key for airport
  airport_name <- incidents$airport[i]
  row =  aid_airport_names[aid_airport_names$airportName == airport_name],]
  if(length(row) > 0){
    incidents$airport[i] = row['aid'][,1]
  } else{
    incidents$airport[i] =NA
  }
   #choose the foreign key for sky_conditions
  sky_conditions <- incidents$sky_conditions[i]
  row =  cid_conditions[cid_conditions$conditions == sky_conditions),]
  incidents$sky_conditions[i] = row['cid'][,1]


  flight_phase = incidents$flight_phase[i]
  if (flight_phase == "Climb"){
    incidents$flight_phase[i] = "takeoff"
  }else if (flight_phase == "Landing Roll"){
    incidents$flight_phase[i] = "landing"
  }else if (flight_phase == "Approach"){
    incidents$flight_phase[i] = "landing"
  }else if (flight_phase == "Take-off run"){
    incidents$flight_phase[i] = "takeoff"
  }else if (flight_phase == "Descent"){
    incidents$flight_phase[i] = "landing"
  }else{
    incidents$flight_phase[i] = "unknown"
  }

  pilot_warned_flag = incidents$pilot_warned_flag[i]
  if (pilot_warned_flag == "N"){
    incidents$pilot_warned_flag[i] = "0"
  }else{
    incidents$pilot_warned_flag[i] = "1"
  }

  incidents$flight_date[i] = as.character(as.Date(incidents$flight_date[i], "%m/%d/%Y %H:%M"))
}
```

#rename columns

Hide

```
colnames(incidents)[colnames(incidents) == "flight_date"] <- "date"
colnames(incidents)[colnames(incidents) == "airport"] <- "origin"
colnames(incidents)[colnames(incidents) == "flight_phase"] <- "flightPhase"
colnames(incidents)[colnames(incidents) == "altitude_ft"] <- "altitude"
colnames(incidents)[colnames(incidents) == "sky_conditions"] <- "conditions"
colnames(incidents)[colnames(incidents) == "pilot_warned_flag"] <- "warning"
```

#incidents # insert to incidents table

Hide

```
dbWriteTable(mydb, "incidents", incidents, override=TRUE, append=TRUE, row.names=FALSE)
```

4. Create a SQL query against your database to find the 10 airlines with the greatest number of incidents

Hide

```
select airline, count(*) as count from incidents group by airline order by count DESC limit 10;
```

5. Create a SQL query against your database to find the flight phase that had an above average number bird strike incidents (during any flight phase).

Hide

```
select A.flightPhase from (SELECT flightPhase, COUNT(*) AS flightPhaseCount FROM incidents
  GROUP BY flightPhase) as A, (SELECT AVG(C.flightPhaseCount) as aver
  from(SELECT COUNT(*) AS flightPhaseCount FROM incidents
  GROUP BY flightPhase) as C) as B where flightPhaseCount > B.aver;
```

6. Create a SQL query against your database to find the number of bird strike incidents by month (across all years). Include all airlines and all flights

Hide

```
SELECT month (date), count(*) FROM incidents
GROUP BY month(date);
```

7. Build a line chart that visualizes the number of bird strikes incidents per year from 2005 to 2011. Adorn the graph with appropriate axis labels, titles, legend, data labels, etc.

Hide

```
rs <- dbGetQuery(mydb, "select count(*) as 'bird_strikes_incidents', year(date) as year FROM incidents
where year(date) between 2005 and 2011
group by year(date);")

plot(x=rs$year, y=rs$bird_strikes_incidents,xlab="year", ylab="bird strikes incidents",main="bird strikes incidents per year from 2005 to 2011")
```

8. Create a stored procedure in MySQL

Hide

```
CREATE PROCEDURE InsertIncidents(
    IN rid int,
    IN date date,
    IN origin int,
    IN airline varchar(50),
    IN aircraft varchar(50),
    IN flightPhase varchar(50),
    IN altitude varchar(50),
    IN conditions int,
    IN warning boolean
)
BEGIN
    INSERT INTO incidents
    (rid,
    date,
    origin,
    airline,
    aircraft,
    flightPhase,
    altitude,
    conditions,
    warning)
VALUES
    (rid,
    date,
    origin,
    airline,
    aircraft,
    flightPhase,
    altitude,
    conditions,
    warning);
END
```

Hide

```
CALL InsertIncidents('200099', '2001-04-06', '48', 'ABX AIR', 'Airplane', 'landing', '0', '3', '1');
```

Hide

```
select * FROM incidents where rid = '200099';
```

#close database

Hide

```
dbDisconnect(mydb)
```