#Creating an Airline table as per data dictionary. Airline stores all airlines and their reputations. Primary Key set to Airline\_code

CREATE TABLE Airline (

#primary Key therefore not null required

Airline\_Code VARCHAR(5) NOT NULL ,

Airline\_Name VARCHAR(40) NOT NULL,

Reputation\_Rate FLOAT(5) NOT NULL ,

Num\_Accidents INT NOT NULL ,

PRIMARY KEY ( Airline\_Code )

) ;

#Creating an Airports table as per data dictionary. Airports table stores data about airports (arrival and departure airport) Primary Key set to Airport\_Code

CREATE TABLE Airports (

#primary Key therefore not null required

Airport\_Code VARCHAR(5) NOT NULL ,

Airport\_Name VARCHAR(30) NOT NULL ,

Airport\_City VARCHAR(30) NOT NULL ,

Airport\_Capacity INT(15) NOT NULL ,

Reputation FLOAT(5) NOT NULL ,

Duty\_Free BOOLEAN NOT NULL ,

Lounge BOOLEAN NOT NULL ,

Average\_Wait\_Time FLOAT(5) NOT NULL ,

PRIMARY KEY ( Airport\_Code )

) ;

#Creating a Passengers table as per data dictionary. Passenger table stores details about passengers like email, age, gender, passport number. Primary Key set to Passenger\_ID

CREATE TABLE Passengers (

#primary Key therefore not null required

Passenger\_ID INT(10) NOT NULL ,

Passport\_Number VARCHAR(10) NOT NULL ,

First\_Name VARCHAR(20) NOT NULL ,

Last\_Name VARCHAR(20) NOT NULL ,

Sex VARCHAR(8) NOT NULL ,

Age INT(3) NOT NULL ,

Birth DATE NOT NULL ,

Phone VARCHAR(20) NOT NULL ,

Email VARCHAR(40) NOT NULL ,

PRIMARY KEY ( Passenger\_ID )

);

#Creating an aircraft table as per data dictionary. Aircraft stores details about aircraft including number of sets. Primary Key set to Model.

# Model corresponds to model of aircraft

CREATE TABLE Aircraft (

#primary Key therefore not null required

Model VARCHAR(20) NOT NULL ,

Manufacturer VARCHAR(20) NOT NULL ,

#The aircraft either supports wifi for all seats or does not

Wifi\_Option BOOLEAN NOT NULL ,

Passenger\_Capacity INT(3) NOT NULL ,

Year\_Build INT(4) NOT NULL ,

PRIMARY KEY ( Model )) ;

#Creating a flight table as per data dictionary. Flight stores details about flights including CO2 emission, arrival and departure locations. Primary Key set to Flight ID.

#Adding Foreign Key Airport\_Code\_Departure from Airport table

#Adding Foreign Key Airport\_Code\_Arrival from Airport table

#Adding Foreign Key Airline\_Code from Airline table

#Adding Foreign Key Model from Aircraft table

CREATE TABLE Flight (

Flight\_ID INT(10) NOT NULL ,

Departure\_Datetime DATETIME(6) NOT NULL ,

Reference\_Price FLOAT(8) NOT NULL ,

Duration FLOAT(5) NOT NULL ,

Arrival\_Datetime DATETIME(6) NOT NULL ,

CO2\_Emissions FLOAT(10) NOT NULL ,

Airline\_Code VARCHAR(5) NOT NULL ,

Airport\_Code\_Departure VARCHAR(5) NOT NULL ,

Airport\_Code\_Arrival VARCHAR(5) NOT NULL ,

Model VARCHAR(20) NOT NULL ,

PRIMARY KEY ( Flight\_ID ),

FOREIGN KEY ( Airport\_Code\_Departure ) REFERENCES Airports ( Airport\_Code ),

FOREIGN KEY ( Airport\_Code\_Arrival ) REFERENCES Airports ( Airport\_Code ),

FOREIGN KEY ( Airline\_Code) REFERENCES Airline ( Airline\_Code ),

FOREIGN KEY ( Model ) REFERENCES Aircraft ( Model )

);

#Creating a Seat table as per data dictionary. Seat stores details about seat features like leg room and which seat a ticket is associated with. Primary Key set to Seat\_ID

#Adding Foreign Key Flight\_ID from Flight table

CREATE TABLE Seat (

#primary Key therefore not null required

Seat\_ID VARCHAR(13) NOT NULL ,

Class VARCHAR(15) NOT NULL ,

Leg\_Space FLOAT(5) NOT NULL ,

#unlike wifi USB can be seat specific

USB BOOLEAN NOT NULL ,

Flight\_ID INT(10) NOT NULL ,

PRIMARY KEY ( Seat\_ID ),

Foreign Key ( Flight\_ID ) REFERENCES Flight (Flight\_ID)

) ;

#Creating a Ticket table as per data dictionary. Ticket stores data about the passengers purchase (meal,discount). Primary Key set to Reference\_Num

#Adding Foreign Key Seat\_ID from Seat table

#Adding Foreign Key Passenger\_ID from Passenger table

CREATE TABLE Ticket (

#primary Key therefore not null required

Reference\_Num INT(10) NOT NULL ,

#Purchase ID required as all tickets a purchase

Purchase\_ID INT(10) NOT NULL ,

Priority BOOLEAN NOT NULL ,

Meal\_Option BOOLEAN NOT NULL ,

Discount FLOAT(5) NOT NULL ,

Purchase\_Date DATE NOT NULL ,

Price\_Paid FLOAT(8) NOT NULL ,

Seat\_ID VARCHAR(13) NOT NULL ,

Passenger\_ID INT(10) NOT NULL ,

PRIMARY KEY ( Reference\_Num ) , FOREIGN KEY( Seat\_ID ) REFERENCES Seat ( Seat\_ID ), FOREIGN KEY( Passenger\_ID ) REFERENCES Passengers ( Passenger\_ID ));

#Airport Data for 8 airports

INSERT INTO `Airports` (`Airport\_Code`, `Airport\_Name`, `Airport\_City`, `Airport\_Capacity`, `Reputation`, `Duty\_Free`, `Lounge`, `Average\_Wait\_Time`)

VALUES

('YUL', 'Trudeau Airport', 'Montreal', '520175', '4.3', '1', '1', '45'),

('YYZ', 'Pearson Airport', 'Toronto', '781971', '4.1', '1', '1', '55'),

('CDG', 'Charles de Gaulle Airport', 'Paris', '648725', '3.8', '1', '1', '33.5'),

('CUN', 'Cancun Airport', 'Cancun', '154962', '4.6', '1', '1', '25'),

('YYG', 'Charlottetown Airport', 'Charlottetown', '3000', '2', '0', '0', '5'),

('LHR','London Heathrow','London','100000','3.9','1','1','64.2'),

('YVR','Vancouver International', 'Vancouver', '35000','2.9','1','0','14'),

('VIE','Vienna International Airport', 'Vienna', '32030','3.3','1','0','27');

#Airline Data for 6 carriers

INSERT INTO `Airline` (`Airline\_Code`, `Airline\_Name`, `Reputation\_Rate`, `Num\_Accidents`) VALUES

('WS', 'WestJet', '3.9', '3'),

('AC', 'Air Canada', '4.7', '3'),

('AG', 'Deutsche Lufthansa', '4.1', '30'),

('AF', 'Air France', '3.5', '48'),

('DL', 'Delta Air Lines', '4', '10'),

('TS','Air Transat' ,'3.8','12');

#Aircraft data for 7 aircraft models

#There are surprisingly few aircraft models

INSERT INTO `Aircraft` (`Model`, `Manufacturer`, `Wifi\_option`, `Passenger\_capacity`, `Year\_Build`)

VALUES

('Boeing 777', 'Boeing', '1', '388', '1994'),

('Airbus A350', 'Airbus', '1', '410', '2016'),

('Airbus A321neo', 'Airbus', '1', '244', '2016'),

('Boeing 737', 'Boeing', '0', '162', '1989'),

('Airbus A321', 'Airbus', '0', '200', '1992'),

('Airbus A319', 'Airbus', '0', '156', '1996'),

('Boeing 787', 'Boeing', '1', '290', '2007');

#50 sample passengers data for database

#Names chose from to 100 popular names

#All other data randomly generated

INSERT INTO Passengers (`Passenger\_ID`,`Passport\_Number`,`First\_Name`,`Last\_Name`,`Sex`,`Age`,`Birth`,`Phone`,`Email`) VALUES('1000000','E533388485','Barbara','Khatun','Female','62','1961/02/09','(103)-623-5729','BarbaraKhatun@gmail.com'),

('1000001','E533389002','Ashley','Jones','Male','42','1980/10/20','(914)-512-5037','AshleyJones@mcgill.ca'),

('1000002','E533389098','Sharon','Gao','Male','29','1994/05/04','(960)-236-6562','SharonGao@msn.com'),

('1000003','E533384139','Susan','Jiang','Female','70','1952/12/03','(480)-226-3276','SusanJiang@yahoo.com'),

('1000004','E533387876','Rebecca','Liu','Female','51','1971/12/27','(531)-944-6134','RebeccaLiu@gmail.com'),

('1000005','E533389365','Betty','Davies','Female','80','1943/04/16','(913)-343-9174','BettyDavies@gmail.com'),

('1000006','E533382471','Nancy','Nguyen','Female','19','2004/02/21','(375)-892-1496','NancyNguyen@yahoo.com'),

('1000007','E533388499','Laura','Guo','Male','75','1948/06/08','(862)-909-6512','LauraGuo@yahoo.com'),

('1000008','E533387004','Betty','Zhang','Male','56','1967/02/26','(940)-402-7856','BettyZhang@yahoo.com'),

('1000009','E533388163','Karen','Ahmed','Male','21','2002/02/22','(567)-257-1807','KarenAhmed@gmail.com'),

('1000010','E533386740','Ashley','Smith','Male','33','1989/11/14','(679)-402-4949','AshleySmith@yahoo.com'),

('1000011','E533383624','Lisa','Jiang','Female','67','1956/02/06','(419)-569-1811','LisaJiang@gmail.com'),

('1000012','E533383982','Jennifer','Zheng','Female','63','1960/03/23','(578)-803-7487','JenniferZheng@yahoo.com'),

('1000013','E533388793','Brenda','Tang','Female','28','1995/06/23','(763)-978-3707','BrendaTang@gmail.com'),

('1000014','E533384451','Mary','Kim','Male','76','1947/06/28','(608)-381-7534','MaryKim@msn.com'),

('1000015','E533388609','Amy','Ali','Female','72','1951/05/06','(968)-280-9589','AmyAli@gmail.com'),

('1000016','E533383161','Amy','Khatun','Female','61','1961/09/28','(127)-453-3588','AmyKhatun@msn.com'),

('1000017','E533389628','Emily','Kumar','Female','71','1952/05/02','(925)-653-8977','EmilyKumar@msn.com'),

('1000018','E533383555','Jessica','Smith','Male','27','1995/11/13','(909)-157-8211','JessicaSmith@msn.com'),

('1000019','E533386148','Jennifer','Huang','Female','30','1992/09/04','(174)-321-6017','JenniferHuang@mcgill.ca'),

('1000020','E533387853','Jennifer','Wei','Female','75','1948/05/09','(527)-112-5852','JenniferWei@yahoo.com'),

('1000021','E533388103','Brenda','Yang','Female','51','1972/03/01','(709)-224-8562','BrendaYang@gmail.com'),

('1000022','E533384753','Nicole','Lu','Female','66','1956/11/10','(865)-170-4495','NicoleLu@msn.com'),

('1000023','E533387471','Margaret','Lin','Female','55','1968/04/02','(453)-288-9509','MargaretLin@yahoo.com'),

('1000024','E533385658','Patricia','Ali','Female','34','1988/11/29','(742)-147-9391','PatriciaAli@gmail.com'),

('1000025','E533388235','Carol','Guo','Female','49','1974/03/24','(242)-276-3580','CarolGuo@gmail.com'),

('1000026','E533388190','Sharon','Wu','Male','54','1969/01/25','(562)-273-7045','SharonWu@gmail.com'),

('1000027','E533384549','Angela','Lu','Male','26','1997/02/01','(409)-280-9715','AngelaLu@yahoo.com'),

('1000028','E533386198','Amanda','Sun','Male','67','1956/04/12','(302)-772-7353','AmandaSun@gmail.com'),

('1000029','E533383699','Angela','Zhao','Female','51','1972/05/08','(256)-267-8534','AngelaZhao@gmail.com'),

('1000030','E533387517','Sandra','Maung','Male','30','1993/05/14','(541)-283-1117','SandraMaung@msn.com'),

('1000031','E533381369','Stephanie','Yang','Male','27','1996/02/02','(418)-804-2048','StephanieYang@gmail.com'),

('1000032','E533385270','Amy','Smith','Female','66','1956/11/05','(757)-191-4521','AmySmith@mcgill.ca'),

('1000033','E533381590','Deborah','Lu','Female','27','1996/06/03','(914)-681-8130','DeborahLu@gmail.com'),

('1000034','E533388234','Pamela','Garcia','Male','23','2000/04/23','(591)-656-4776','PamelaGarcia@msn.com'),

('1000035','E533385456','Pamela','Nguyen','Female','54','1969/06/02','(910)-437-5007','PamelaNguyen@gmail.com'),

('1000036','E533387216','Anna','Lu','Male','61','1961/10/13','(901)-939-4163','AnnaLu@gmail.com'),

('1000037','E533386838','Susan','Liu','Male','80','1943/04/07','(972)-431-7429','SusanLiu@gmail.com'),

('1000038','E533385121','Helen','Kumar','Female','31','1991/11/24','(926)-341-5730','HelenKumar@Oulook.com'),

('1000039','E533389346','Jennifer','Liang','Female','21','2001/09/16','(358)-528-2553','JenniferLiang@gmail.com'),

('1000040','E533382619','Amanda','Kumar','Female','75','1947/11/02','(152)-732-7028','AmandaKumar@gmail.com'),

('1000041','E533387829','Carol','Maung','Female','54','1969/06/15','(992)-889-1390','CarolMaung@yahoo.com'),

('1000042','E533381552','Sharon','Devi','Male','45','1977/11/12','(845)-295-8259','SharonDevi@gmail.com'),

('1000043','E533385588','Michelle','dailva','Female','50','1973/06/20','(152)-357-8755','Michelleda Silva@gmail.com'),

('1000044','E533387268','Barbara','Huang','Female','21','2001/09/19','(622)-261-5160','BarbaraHuang@gmail.com'),

('1000045','E533383158','Dorothy','Lu','Female','30','1993/02/05','(251)-487-2569','DorothyLu@yahoo.com'),

('1000046','E533382412','Nancy','Zheng','Male','53','1970/04/29','(175)-657-5890','NancyZheng@msn.com'),

('1000047','E533383005','Lisa','Jiang','Female','48','1975/05/26','(890)-404-5590','LisaJiang@yahoo.com'),

('1000048','E533382695','Melissa','Davies','Female','21','2001/09/11','(546)-466-8055','MelissaDavies@gmail.com');

#Add 11 sample flights

INSERT INTO Flight (`Flight\_ID`, `Departure\_Datetime`, `Reference\_Price`, `Duration`, `Arrival\_Datetime`, `CO2\_Emissions`, `Airline\_Code`, `Airport\_Code\_Departure`, `Airport\_Code\_Arrival`, `Model`)

VALUES

('1000000000', '2021-09-07 02:30:00', '778', '7', '2021-09-07 09:30:00', '383', 'AC', 'YYZ', 'LHR', 'Boeing 777'),

('1000000001', '2021-10-11 17:15:00', '813', '7.25', '2021-10-12 00:45:00', '301', 'AF', 'YYZ', 'CDG', 'Airbus A350'),

('1000000002', '2021-10-23 13:00:00', '759', '8.5', '2021-10-23 21:45:00', '338', 'TS', 'CDG', 'YYZ', 'Airbus A321neo'),

('1000000003', '2021-11-01 15:30:00', '408', '4', '2021-11-01 17:30:00', '172', 'WS', 'YYZ', 'CUN', 'Boeing 737'),

('1000000004', '2021-11-20 07:00:00', '507', '4.5', '2021-11-20 11:00:00', '211', 'AC', 'CUN', 'YUL', 'Airbus A321'),

('1000000005', '2021-12-13 22:15:00', '167', '1.5', '2021-12-13 23:45:00', '86', 'AC', 'YUL', 'YYG', 'Airbus A319'),

('1000000006', '2022-01-26 11:15:00', '1413', '9', '2022-01-26 20:30:00', '380', 'AC', 'VIE', 'YYZ', 'Boeing 787'),

('1000000007', '2022-02-11 09:00:00', '329', '4.5', '2022-02-11 13:00:00', '254', 'WS', 'YVR', 'YYZ', 'Boeing 737'),

('1000000008', '2022-03-07 20:30:00', '874', '9.75', '2022-03-08 05:45:00', '506', 'AF', 'YVR', 'CDG', 'Boeing 777'),

('1000000009', '2022-04-26 14:15:00', '684', '6', '2022-04-26 20:30:00', '430', 'AG', 'YUL', 'CUN', 'Boeing 787'),

('1000000010', '2022-05-12 01:00:00', '984', '7.5', '2022-05-12 09:00:00', '238', 'DL', 'YVR', 'CDG', 'Airbus A321');

#Add 50 sample seats, with randomly assigned passengers

INSERT INTO Seat (`Seat\_ID`,`Class`,`Leg\_space`,`USB`,`Flight\_ID`)

VALUES

('6E1000000000','Economy','35','0','1000000000'),

('19E1000000001','First','51','1','1000000001'),

('3B1000000002','Economy','29','0','1000000002'),

('15B1000000003','Economy','31','0','1000000003'),

('17E1000000004','Business','43','1','1000000004'),

('6D1000000005','Economy','32','1','1000000005'),

('27C1000000006','Economy','35','0','1000000006'),

('4F1000000007','Economy','35','0','1000000007'),

('12E1000000008','First','55','0','1000000008'),

('18B1000000009','Business','43','0','1000000009'),

('28B1000000010','Economy','33','1','1000000010'),

('28A1000000010','First','52','1','1000000010'),

('8D1000000000','Business','44','0','1000000000'),

('10D1000000001','First','52','0','1000000001'),

('24C1000000002','Business','47','0','1000000002'),

('1B1000000003','Business','44','0','1000000003'),

('1D1000000004','First','49','0','1000000004'),

('17B1000000005','First','55','0','1000000005'),

('17D1000000006','First','56','1','1000000006'),

('21C1000000007','Business','42','1','1000000007'),

('17F1000000008','Economy','35','1','1000000008'),

('2A1000000009','Business','50','0','1000000009'),

('15B1000000010','Economy','30','0','1000000010'),

('23D1000000010','Economy','31','1','1000000010'),

('8F1000000000','Business','50','1','1000000000'),

('19C1000000001','Economy','35','0','1000000001'),

('9A1000000002','Economy','34','1','1000000002'),

('21C1000000003','First','55','1','1000000003'),

('20F1000000004','Economy','34','1','1000000004'),

('0A1000000005','Business','42','0','1000000005'),

('7A1000000006','First','49','1','1000000006'),

('11A1000000007','Economy','32','0','1000000007'),

('23E1000000008','Business','44','1','1000000008'),

('3E1000000009','First','58','0','1000000009'),

('17F1000000010','First','50','0','1000000010'),

('22F1000000010','Business','40','1','1000000010'),

('20E1000000000','First','57','1','1000000000'),

('26F1000000001','Economy','34','0','1000000001'),

('23A1000000002','First','53','1','1000000002'),

('18B1000000003','Economy','32','0','1000000003'),

('3B1000000004','Economy','33','1','1000000004'),

('29E1000000005','First','53','0','1000000005'),

('4A1000000006','Economy','27','1','1000000006'),

('20E1000000007','Business','42','1','1000000007'),

('10E1000000008','Business','43','0','1000000008'),

('21B1000000009','Economy','34','1','1000000009'),

('22C1000000010','First','49','1','1000000010'),

('28C1000000010','First','52','1','1000000010'),

('28D1000000000','Business','51','1','1000000000');

#Add 50 sample data to tickets table for each passenger

INSERT INTO Ticket (`Reference\_Num`,`Purchase\_ID`,`Priority`,`Meal\_Option`,`Discount`,`Purchase\_Date`,`Price\_Paid`,`Seat\_ID`,`Passenger\_ID`) VALUES ('98116678','100000','5','1','0.58','2021/07/28','2123','6E1000000000','1000047'),

('25288789','100001','5','0','0.13','2021/09/03','141','19E1000000001','1000009'),

('15424738','100002','3','1','0.04','2021/04/12','3372','3B1000000002','1000039'),

('76565625','100003','3','0','0.75','2022/05/31','2068','15B1000000003','1000033'),

('88946875','100004','2','0','0.54','2021/03/18','3416','17E1000000004','1000033'),

('68625682','100005','2','0','0.16','2021/08/29','2063','6D1000000005','1000026'),

('97789717','100006','2','1','0.54','2021/11/27','2353','27C1000000006','1000008'),

('10115493','100007','3','0','0.36','2021/01/20','3141','4F1000000007','1000011'),

('85081387','100008','3','1','0.44','2022/07/25','3528','12E1000000008','1000028'),

('83649941','100009','5','1','0.67','2021/11/28','1973','18B1000000009','1000014'),

('46762687','100010','5','1','0.43','2021/02/03','2660','28B1000000010','1000045'),

('67971328','100011','4','0','0.11','2021/09/06','1646','28A1000000010','1000038'),

('58979937','100012','3','1','0.75','2022/07/21','2767','8D1000000000','1000010'),

('26981698','100013','3','0','0.66','2021/06/07','3825','10D1000000001','1000006'),

('33677835','100014','4','1','0.55','2022/07/26','2820','24C1000000002','1000023'),

('70208103','100015','3','1','0.51','2021/12/09','1155','1B1000000003','1000005'),

('98917854','100016','1','0','0.56','2022/05/08','3190','1D1000000004','1000031'),

('60310346','100017','1','0','0.53','2021/06/22','2786','17B1000000005','1000041'),

('98278906','100018','5','0','0.62','2022/06/29','459','17D1000000006','1000008'),

('53265728','100019','1','1','0.25','2021/09/25','1892','21C1000000007','1000034'),

('10488723','100020','4','0','0.05','2021/05/24','1942','17F1000000008','1000022'),

('76812966','100021','4','0','0.49','2021/09/14','434','2A1000000009','1000034'),

('50649546','100022','5','0','0.03','2021/05/08','2900','15B1000000010','1000048'),

('66210951','100023','3','1','0.2','2021/07/20','1595','23D1000000010','1000023'),

('61429562','100024','2','1','0.38','2022/03/28','3115','8F1000000000','1000002'),

('42127317','100025','3','1','0.15','2021/08/19','2922','19C1000000001','1000022'),

('81167276','100026','3','0','0.12','2021/07/03','2488','9A1000000002','1000041'),

('89932920','100027','2','1','0.56','2022/02/21','1835','21C1000000003','1000048'),

('34661417','100028','1','1','0.12','2022/04/02','709','20F1000000004','1000025'),

('99413467','100029','4','0','0.05','2021/06/11','871','0A1000000005','1000007'),

('56581962','100030','4','0','0.61','2021/05/15','2149','7A1000000006','1000003'),

('18147218','100031','5','0','0.32','2022/07/02','1729','11A1000000007','1000003'),

('10932396','100032','1','1','0.54','2022/01/14','2923','23E1000000008','1000022'),

('84922763','100033','2','0','0.26','2021/11/13','741','3E1000000009','1000012'),

('28220074','100034','4','1','0.49','2022/05/03','2279','17F1000000010','1000033'),

('33624220','100035','4','1','0.69','2021/05/07','3743','22F1000000010','1000024'),

('74256816','100036','2','1','0.25','2021/11/19','2881','20E1000000000','1000044'),

('15190210','100037','4','0','0.09','2022/02/04','1742','26F1000000001','1000018'),

('96528908','100038','1','1','0.18','2022/07/06','1744','23A1000000002','1000038'),

('40411108','100039','3','0','0.7','2022/08/22','2545','18B1000000003','1000018'),

('10463257','100040','2','0','0.63','2022/03/07','714','3B1000000004','1000046'),

('29476021','100041','4','0','0.08','2021/09/08','2544','29E1000000005','1000001'),

('34700329','100042','2','0','0.54','2021/09/14','845','4A1000000006','1000013'),

('92982920','100043','2','1','0.13','2022/06/29','136','20E1000000007','1000009'),

('11595972','100044','1','1','0.21','2022/03/06','1430','10E1000000008','1000036'),

('11404421','100045','3','0','0.13','2021/11/10','3179','21B1000000009','1000017'),

('52951489','100046','2','1','0.6','2022/05/02','3364','22C1000000010','1000016'),

('26236480','100047','5','1','0.45','2022/05/14','3860','28C1000000010','1000015'),

('36658249','100048','5','1','0.37','2022/03/12','3832','28D1000000000','1000034');

#Query 1

#The purpose of this query is to understand if the duration of the flight is correlated to the amount of CO2 emitted by the aircraft in order to better meet the demand of eco-friendly travelers.

SELECT Flight\_ID, Duration, CO2\_Emissions

FROM Flight

WHERE Duration < 5;

#Query 2

#The purpose of this question is for the team to know what aircrafts they should not use for their flights as they are more ancient and do not have the WIFI option.

SELECT Model,

COUNT(Model) AS Number\_Aircraft\_No\_Wifi,

Wifi\_Option

FROM Aircraft

WHERE Year\_Build < 2008

AND Wifi\_Option = 0

#Displays aircraft model and allows query results to be grouped by aircraft model

GROUP BY Model;

#Query 3

#The purpose of this query is for the airline managers to understand who are the most popular airlines on the market that significantly stand out from the others.

SELECT Airline\_Name,

Reputation\_rate

FROM Airline

WHERE Reputation\_rate > (SELECT AVG (Reputation\_rate) AS average\_rep FROM Airline);

#Query 4

#The purpose of this query is for the marketing manager to target the most popular airports and to understand more in depth where this high rate comes from.

SELECT Airport\_name, Reputation

FROM Airports

WHERE Reputation > 4;

#Query 5

#The marketing manager wants to launch a new ad campaign that would ultimately increase our customer retention. He needs the average age of our customers in order to create a campaign that would be tailored to the specific demographic of our customers.

SELECT Sex,

AVG(Age) AS average\_age

FROM Passengers

GROUP BY Sex;

#Query 6

#The customers who have ever bought a ticket whose Price\_Paid is greater than $2000 are considered “top-tier” customers. Show how many “top-tier” customers the company has.

SELECT DISTINCT COUNT(Passenger\_ID) AS Number\_of\_Top\_Tier

FROM Ticket

WHERE Price\_Paid >= 2000;

#Query 7

#Our shareholders are becoming increasingly concerned by the environment. They want us to display the average CO2 emission of all the flights in our database in order to see if our business is environmentally friendly.

SELECT AVG(CO2\_Emissions) AS Average\_CO2\_Emission

FROM Flight;

#Query 8

#We want to improve the customer experience of our top-tier customers, as a result we will recommend them flights that are going to the best rated airports.

SELECT Airport\_Name, Reputation

FROM Airports

#Subquery allows selection of average for all values. Otherwise average would be average per row aka the reputation value

WHERE Reputation > (SELECT AVG (Reputation) AS average\_rep FROM Airports);

#Query 9

#Customers are becoming more price sensitive. We want to determine the range of prices our site offers to update our rating on google. Keeping our prices low will allow us to remain competitive.

SELECT MAX(Price\_Paid) AS Max\_Price,

Min(Price\_Paid) AS Minimum\_Price

FROM Ticket;

#Query 10

#We would like to improve our offerings for Gmail account users to make our emails go from the “promotion” section to the “primary section” as sales believes this will give us better visibility, but only for Women. How many Female customers do we currently have on gmail?

SELECT

a.Sex,

COUNT(Sex) AS "Number of Gmail Emails"

FROM

#The subquery pulls together a table of all male and female email addresses.

(SELECT

p.Passenger\_ID,

p.Sex

#Cartesian joining together the passengers and ticket

FROM Passengers p, Ticket t

#Joining on primary key similarity

WHERE p.Passenger\_ID = t.Passenger\_ID

#The “LIKE” allows searching with wildcard as needed here

AND p.Email LIKE "%@gmail.com") a

GROUP BY Sex;

#Query 11

#We are looking to grow our business by advertising for our least popular destinations. What is the least popular destination we have booked and can use for advertising?

SELECT

a.Airport\_City AS "Destination",

COUNT(a.Airport\_City) AS "Popularity"

FROM

#Subquery that creates larger dataset joining all the tables together needed as group by is used

(SELECT ap.Airport\_City

FROM Ticket t, Seat s, Flight f, Airports ap

WHERE t.Seat\_ID = s.Seat\_ID

#Make sure the join is for the appropriate primary and foreign keys

AND s.Flight\_ID = f.Flight\_ID

AND ap.Airport\_Code=f.Airport\_Code\_Arrival) a

GROUP BY a.Airport\_City

#Sort by popularity descending. Smallest on top as ASC is default sort order

ORDER BY COUNT(a.Airport\_City)

#Limit to top 10

LIMIT 10;

#Query 12

#All airlines are looking for help. They have asked what is our least popular destination so that they can evaluate and either advertise or drop the flights to that destination.

#Query returns a table of all the arrival airports with the number of tickets booked

SELECT

DISTINCT ap.Airport\_City,

ap.Airport\_Code,

COUNT(t.Reference\_Num)

#Cartesian joining together the ticket, seat, flight and airports tables

FROM Ticket t, Seat s, Flight f, Airports ap

#Make sure the join is for the appropriate primary and foreign keys

WHERE t.Seat\_ID = s.Seat\_ID

AND s.Flight\_ID = f.Flight\_ID

AND ap.Airport\_Code = f.Airport\_Code\_Arrival

GROUP BY ap.Airport\_Code

#Sort by popularity ASC. Least popular on top

ORDER BY COUNT(ap.Airport\_City) ASC;

#Query 13

#The airlines are back. They were so impressed with work they wanted help! Now they are worried about their aircraft, and they want help with which aircraft they should retire based on passenger popularity.What is the least popular aircraft based on the average number of tickets per flight?

SELECT

f.model,

ROUND(a.SeatCount/COUNT(f.model),1) AS "Average occupancy"

FROM Flight f, (SELECT

DISTINCT af.model,

COUNT(s.Seat\_ID) AS SeatCount

#Cartesian joining together the aircraft, airlines and flight tables

FROM Aircraft af, Airline al, Flight f, Seat s

WHERE af.Model = f.Model

#Make sure the join is for the appropriate primary and foreign keys

AND al.Airline\_Code = f.Airline\_Code

AND s.Flight\_ID = f.Flight\_ID

GROUP BY af.model

ORDER BY COUNT(af.model) DESC) a

WHERE a.model = f.model

#Group by the model of aircraft

GROUP BY f.model

#Order Average occupancy

ORDER BY a.SeatCount/COUNT(f.model) ASC;

#Query 14

#We are attempting to have an email campaign. The colors and designs are customized based on the customer but some fundamental design choices are biased towards one gender or another and show up better on one domain (Gmail vs. Yahoo) versus others. Who should the marketing team design the email campaign for?

#Select the needed columns for the output

SELECT

p.Sex,

#The below returns the domain name by searching for the first ‘@’ symbol and returning everything to the right of it aka the bit we care about

RIGHT(email, length(email)-INSTR(email, '@')),

100\*COUNT(RIGHT(email, length(email)-INSTR(email, '@')))/c.Total\_Count AS "% of Email Domaine"

#Cartesian joining together the passengers and custom c table

FROM Passengers p,

#The subquery will pull together a total count of all rows needed for proportion

#This is then cartesian joined to the other results but grouped by is applied

(SELECT

COUNT(\*) AS "Total\_Count"

FROM Passengers) c

#Grouping the emails by domain

GROUP BY RIGHT(email, length(email)-INSTR(email, '@')),c.Total\_Count,p.Sex

ORDER BY COUNT(RIGHT(email, length(email)-INSTR(email, '@'))) DESC;

#Query 15

#In recent years, increasing global warming has put increasing pressure on the aviation industry. Green aviation has become a hot topic, and both governments and ICAO attach great importance to it. So, we want to look at the average CO2 emissions of flights for the specific aircraft model with the most common flight duration between 5 and 10 hours, to provide data support for our subsequent optimization decisions. For now we will concentrate on the top 3.

SELECT Model, AVG(CO2\_Emissions) AS average

FROM Flight

#Flight duration between 5 & 10 hours

WHERE Duration BETWEEN 5 AND 10

GROUP BY Model

ORDER BY average DESC

#Only want to investigate the top 3 for now

LIMIT 3;

#Query 16

#Recently, aircraft accidents have become more and more frequent. We want to know if each airline's reputation score would be affected by these incidents, so that better decisions could be made to improve the reputation rate.

SELECT r,

(CASE WHEN ABS(r) BETWEEN 0.9 AND 1.0 THEN "Very highly correlated"

WHEN ABS(r) BETWEEN 0.7 AND 0.9 THEN "Highly correlated"

WHEN ABS(r) BETWEEN 0.5 AND 0.7 THEN "Moderately correlated"

WHEN ABS(r) BETWEEN 0.3 AND 0.5 THEN "Low correlated"

ELSE 'No significant correlation' END) AS CorrelationCoefficients

FROM(

#The subquery returns a table with the correlation coefficient value, the following formula for calculating the correlation coefficient can be simplified to: SUM( ( x - @x ) \* (y - @y) ) / ( (count(x) - 1) \* @div ).

SELECT

ROUND((SUM(

#(x - @x)

#@x means average of x

(Reputation\_Rate - (SELECT ROUND(AVG(Reputation\_Rate),2) AS Avg\_Reputation\_Rate FROM Airline))

#\* (y - @y)

#@y means average of y

\* (Num\_Accidents - (SELECT ROUND(AVG(Num\_Accidents),2) AS Avg\_Num\_Accidents FROM Airline))

)

/ (

#count(x) - 1

(COUNT(Reputation\_Rate) - 1)

#\* @div

#@div means division, it is calculated by the built-in function, we can simplify it to @div = (STDDEV\_SAMP(x) \* STDDEV\_SAMP(y))

\* (SELECT ROUND(STDDEV\_SAMP(Reputation\_Rate) \* STDDEV\_SAMP(Num\_Accidents),2) AS Division FROM Airline)

)

),2) AS r

FROM Airline

) AS temp;

#Query 17

#We want to improve the passenger experience, so we decided to compare each seat’s leg space with its related average leg space based on its class type and aircraft model. Despite being in the same class of the same aircraft model, leg space can vary depending on the position of the seats in the cabin. So, for those seats that are lower than their corresponding averages, we want to adjust them.

SELECT \*

#Subquery the seat table to get the average seat leg room and join to each flight

FROM(

SELECT Seat\_ID,

Class,

Model,

Leg\_Space,

ROUND(AVG(Leg\_Space) OVER (PARTITION BY Class, Model),2) AS avg\_Leg\_Space

FROM Seat s

JOIN Flight f ON s.Flight\_ID = f.Flight\_ID

ORDER BY Seat\_ID) AS temp

WHERE Leg\_Space < avg\_Leg\_Space;

#Query 18

#There is a customer who lives in Toronto and works on an intensive project, but now, he will have a month break in 2021-Oct and he is willing to travel somewhere by taking flight, but he has no plan right now. Our business website is providing a function to let him search all the eligible flights for his demand by using his city airport and show several important information about the flights. For example, this guy is price sensitive, then we sort the price of each flight for him to let him make a choice.

SELECT Airport\_City AS 'Destination',

Departure\_Datetime,

Reference\_Price,

Duration

FROM Flight AS f

#Joining all the airport and flight data

INNER JOIN Airports AS ap ON f.Airport\_Code\_Departure = ap.Airport\_Code

WHERE ap.Airport\_City = 'Toronto'

AND YEAR (f.Departure\_Datetime) = 2021

#Month function return number month of datetime. 10 is october

AND MONTH (f.Departure\_Datetime) = 10

ORDER BY Reference\_Price ASC;

#Query 19

#For our VIP customers, they are always looking for a high quality fly experience. Especially for businessmen who often need to travel by air for work, we select all the high-quality flights for them that make sure their business won’t be affected.

SELECT al.Airline\_Name,

COUNT(f.Flight\_ID)

FROM Airline AS al

#Inner join flight, aircraft , airports and seat

INNER JOIN Flight AS f

INNER JOIN Aircraft AS ac

INNER JOIN Seat AS s

INNER JOIN Airports as ap ON al.Airline\_Code = f.Airline\_Code

AND f.Model = ac.Model

AND f.Flight\_ID = s.Flight\_ID

AND f.Airport\_Code\_Departure = ap.Airport\_Code

#Only keep first class seats that have a usb, wifi and lounge access

WHERE s.Class = 'First'

AND s.USB = 1

AND ac.Wifi\_Option = 1

AND ap.Lounge = 1

GROUP BY al.Airline\_Code;

#Query 20

#For those customers who have a potential purchase chance, our business is thinking about sending each of them a coupon through SMS message to reactivate them, but due to budget control, we will only send coupons on the birthday of those customers.

SELECT CONCAT(First\_Name,' ', Last\_Name) AS 'Target name',

Birth,

Phone

FROM Passengers AS p

#Subquery allows selection passenger\_ID

WHERE p.Passenger\_ID NOT IN (SELECT Passenger\_ID FROM Ticket);

#Query 21

#For the only flight with the lowest number of accidents, we will advertise this feature on the homepage of the website to attract safety-conscious passengers to check all flights belonging to this airline.

SELECT Flight\_ID,

al.Num\_Accidents AS 'Num\_accident of Airline'

FROM Flight as f

#Joining airline and flight together

INNER JOIN Airline as al ON f.Airline\_Code = al.Airline\_Code

ORDER BY al.Num\_Accidents ASC;