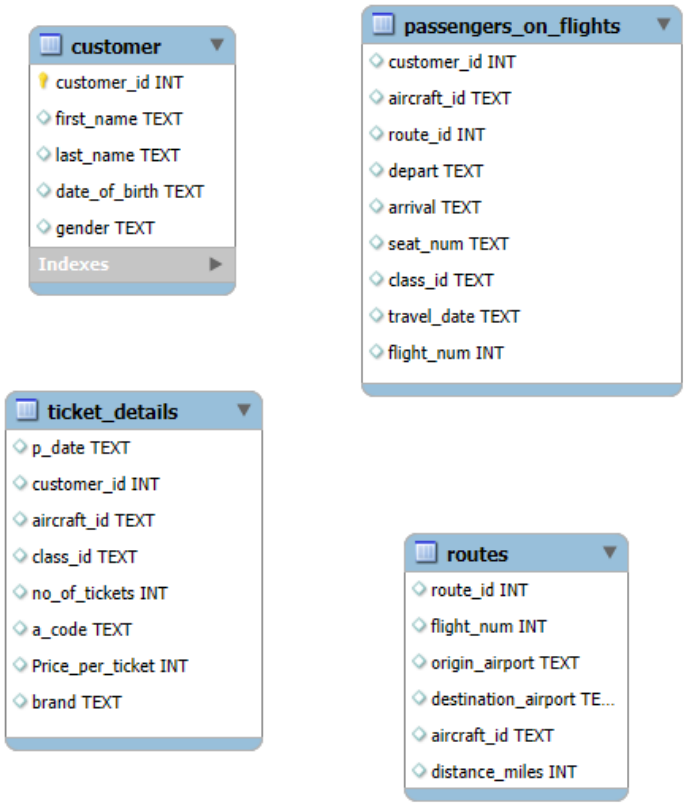
COURSE END PROJECT 2 - AIR CARGO ANALYSIS

CREATE Database air_cargo;

USE air_cargo;

1. Create an ER diagram for the given airlines database.



2. Write a query to create route_details table using suitable data types for the fields, such as route_id, flight_num, origin_airport, destination_airport, aircraft_id, and distance_miles. Implement the check constraint for the flight number and unique constraint for the route_id fields. Also, make sure that the distance miles field is greater than 0.

```
CREATE TABLE        route_details (
                    route_id INT PRIMARY KEY,
                    flight_num INT NOT NULL,
                    origin_airport VARCHAR(50),
                    destination_airport VARCHAR(50),
                    aircraft_id VARCHAR(50) NOT NULL,
                    distance_miles INT CHECK (distance_miles > 0),
                    unique(route_id)
);
```

Field	Type	Null	Key	Default	Extra
route_id	int	NO	PRI	NULL	
flight_num	int	NO		NULL	
origin_airport	varchar(50)	YES		NULL	
destination_airport	varchar(50)	YES		NULL	
aircraft_id	varchar(50)	NO		NULL	
distance_miles	int	YES		NULL	

3. Write a query to display all the passengers (customers) who have travelled in routes 01 to 25. Take data from the passengers_on_flights table.

```
SELECT      *
FROM        passengers_on_flights
WHERE       route_id BETWEEN 01 AND 25
ORDER BY    route_id ASC;
```

customer_id	aircraft_id	route_id	depart	arrival	seat_num	class_id	travel_date	flight_num
18	767-301ER	1	EWR	HNL	13FC	First Class	2018-04-01	1111
2	767-301ER	4	JFK	LAX	01E	Economy	2018-09-02	1114
4	767-301ER	4	JFK	LAX	03FC	First Class	2020-04-30	1114
11	767-301ER	4	JFK	LAX	05B	Bussiness	2020-11-09	1114
4	767-301ER	5	LAX	JFX	02FC	First Class	2020-04-06	1115
11	767-301ER	5	LAX	JFX	04B	Bussiness	2020-11-12	1115
46	A321	8	ORD	EWR	12FC	First Class	2011-07-08	1118
1	ERJ142	9	DEN	LAX	01EP	Economy Plus	2019-12-26	1119
29	ERJ142	9	DEN	LAX	11B	Bussiness	2018-05-03	1119
10	A321	10	HNL	DEN	05E	Economy	2020-10-11	1120
5	767-301ER	12	ABI	ADK	02B	Bussiness	2018-07-02	1122
17	A321	13	ABI	ADK	04EP	Economy Plus	2019-06-03	1123
13	A321	13	ADK	BQN	06FC	First Class	2019-01-05	1123
15	A321	14	BQN	CAK	06B	Bussiness	2018-11-02	1124
24	A321	14	BQN	CAK	08B	Bussiness	2019-07-22	1124
9	767-301ER	15	CAK	ANI	04FC	First Class	2020-09-10	1125
44	767-301ER	15	CAK	ANI	11FC	First Class	2020-10-06	1125
49	767-301ER	15	CAK	ANI	13B	Bussiness	2020-08-19	1125
5	ERJ142	18	ANI	BGR	02E	Economy	2020-05-06	1128
7	767-301ER	20	AVL	BOI	03B	Bussiness	2020-07-08	1130
31	767-301ER	20	AVL	BOI	13E	Economy	2018-12-31	1130
50	A321	21	BFL	BET	10EP	Economy Plus	2020-08-15	1131
5	ERJ142	22	BGR	BJI	03E	Economy	2020-05-31	1132
22	ERJ142	22	BGR	BJI	07EP	Economy Plus	2020-02-09	1132
25	767-301ER	23	BLV	BFL	09B	Bussiness	2019-03-07	1133
46	A321	25	RDM	BJI	14E	Economy	2020-11-25	1135

4. Write a query to identify the number of passengers and total revenue in business class from the ticket_details table.

```
SELECT      count(*) AS bus_pass_num,
            sum(price_per_ticket) AS total_revenue
FROM        ticket_details
WHERE       class_id = 'Bussiness';
```

bus_pass_num	total_revenue
13	6034

5. Write a query to display the full name of the customer by extracting the first name and last name from the customer table.

```
SELECT      concat(first_name, ' ', last_name) AS full_name
FROM        customer;
```

full_name
Julie Sam
Steve Ryan
Morris Lois
Cathenna Emily
Aaron Kim
Alexander Scot
Anderson Stewart
Floyd Ted
Leo Travis
Melvin Tracy
Roger Walson
Shirley Wally
Solomon Walter
Carol Vernon

6. Write a query to extract the customers who have registered and booked a ticket. Use data from the customer and ticket_details tables.

```
SELECT      distinct td.customer_id, c.first_name, c.last_name
FROM        customer c
INNER JOIN  ticket_details td
ON          c.customer_id = td.customer_id
ORDER BY    td.customer_id;
```

customer_id	first_name	last_name
1	Julie	Sam
2	Steve	Ryan
4	Cathenna	Emily
5	Aaron	Kim
7	Anderson	Stewart
8	Floyd	Ted
9	Leo	Travis
10	Melvin	Tracy
11	Roger	Walson
13	Solomon	Walter
14	Carol	Vernon
15	Linda	William
16	Chirstine	Willis
17	Catherine	Shad
18	Gloria	Richie
19	Joyce	Paul
20	Sara	Oliver
21	Chirsty	Josh
22	Pheny	Eri
24	Calvin	Willis
25	Moss	Morris
27	Cherly	Vernon
28	Du plesis	Chris
29	Watson	Ronald
31	James	Robert
32	Chirstoper	Sean
33	Mark	Ethan
41	Kyle	Mark
44	Bily	Brian
46	Louis	Douglas
47	Sophia	Carl
49	Russell	Peter
50	Rose	Arthur

7. Write a query to identify the customer's first name and last name based on their customer ID and brand (Emirates) from the ticket_details table.

```
SELECT      DISTINCT td.customer_id, c.first_name,
             c.last_name, td.brand
FROM        customer c INNER JOIn ticket_details td
On          c.customer_id = td.customer_id
Having      brand = 'Emirates';
```

customer_id	first_name	last_name	brand
27	Cherly	Vernon	Emirates
4	Cathenna	Emily	Emirates
7	Anderson	Stewart	Emirates
9	Leo	Travis	Emirates
11	Roger	Walson	Emirates
25	Moss	Morris	Emirates
18	Gloria	Richie	Emirates
14	Carol	Vernon	Emirates
19	Joyce	Paul	Emirates
5	Aaron	Kim	Emirates
2	Steve	Ryan	Emirates
31	James	Robert	Emirates
49	Russell	Peter	Emirates
44	Bily	Brian	Emirates

8. Write a query to identify the customers who have travelled by Economy Plus class using Group By and Having clause on the passengers_on_flights table.

```
SELECT      customer_id, class_id
FROM        passengers_on_flights
GROUP BY    class_id, customer_id
HAVING      class_id = 'Economy Plus';
```

customer_id	class_id
1	Economy Plus
8	Economy Plus
11	Economy Plus
17	Economy Plus
19	Economy Plus
22	Economy Plus
32	Economy Plus
47	Economy Plus
50	Economy Plus

9. Write a query to identify whether the revenue has crossed 10000 using the IF clause on the ticket_details table.

rev	Rev_check
15369	Crossed

```
SELECT      sum(price_per_ticket) AS rev,
            IF(sum(price_per_ticket) > 10000, 'Crossed', 'Not Crossed') AS Rev_check
FROM        ticket_details;
```

10. Write a query to create and grant access to a new user to perform operations on a database.

```
CREATE USER  'user1' @'localhost'
IDENTIFIED BY  'Password';

GRANT ALL PRIVILIGES ON air_cargo TO 'user1' @'localhost' ;
```

11. Write a query to find the maximum ticket price for each class using window functions on the ticket_details table.

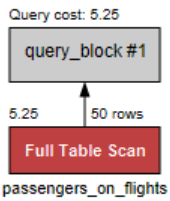
```
SELECT      DISTINCT class_id, max(price_per_ticket) OVER (partition by class_id) AS max_tkt_price
FROM        ticket_details;
```

class_id	max_tkt_price
Bussiness	510
Economy	190
Economy Plus	295
First Class	395

12. Write a query to extract the passengers whose route ID is 4 by improving the speed and performance of the passengers_on_flights table.

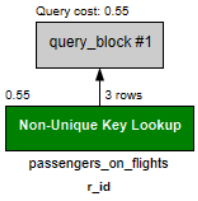
-- to check the current cost and speed

```
SELECT      customer_id, route_id
FROM        passengers_on_flights
WHERE       route_id = 4;
```



-- to check the speed and performance after the index

```
CREATE      index r_id
ON          passengers_on_flights (route_id, customer_id);
ELECT       customer_id, route_id
FROM        passengers_on_flights
WHERE       route_id = 4;
```



customer_id	route_id
2	4
4	4
11	4

13. For the route ID 4, write a query to view the execution plan of the passengers_on_flights table.

```
EXPLAIN SELECT    customer_id, route_id
FROM              passengers_on_flights
WHERE             route_id = 4;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	passengers_on_flights	NULL	ref	r_id	r_id	5	const	3	100.00	Using index

14. Write a query to calculate the total price of all tickets booked by a customer across different aircraft IDs using rollup function.

```
SELECT      customer_id, sum(price_per_ticket) AS total_ticket_price
FROM        ticket_details
GROUP BY    customer_id WITH ROLLUP;
```

customer_id	total_ticket_price
20	680
21	490
22	220
24	480
25	649
27	130
28	170
29	920
31	130
32	220
33	490
41	395
44	380
46	530
47	225
49	430
50	275
NULL	15369

15. Write a query to create a view with only business class customers along with the brand of airlines.

```
CREATE VIEW bus_customers AS

        SELECT      t.customer_id, c.first_name,c.last_name, t.brand
        FROM        ticket_details t INNER JOIN customer c
        ON          t.customer_id = c.customer_id
        WHERE       class_id = 'Bussiness';
```

customer_id	first_name	last_name	brand
2	Steve	Ryan	Qatar Airways
5	Aaron	Kim	Emirates
7	Anderson	Stewart	Emirates
11	Roger	Walson	Emirates
11	Roger	Walson	Emirates
15	Linda	William	Qatar Airways
21	Chirsty	Josh	Bristish Airways
24	Calvin	Willis	Qatar Airways
25	Moss	Morris	Emirates
29	Watson	Ronald	Jet Airways
29	Watson	Ronald	Qatar Airways
33	Mark	Ethan	Bristish Airways
49	Russell	Peter	Emirates

16. Write a query to create a stored procedure to get the details of all passengers flying between a range of routes defined in run time. Also, return an error message if the table doesn't exist.

```
USE `air_cargo`;
DROP procedure IF EXISTS `get_passenger_details`;

DELIMITER //
CREATE PROCEDURE get_passenger_details(IN start_route VARCHAR(50), IN end_route VARCHAR(50))
BEGIN
DECLARE     table_exists INT;

-- Check if the passenger_details table exists

SELECT      COUNT(*) INTO table_exists
FROM        passengers_on_flights;

  -- If table does not exist, return an error message

  IF table_exists = 0 THEN
                    SIGNAL SQLSTATE '45000'
                    SET MESSAGE_TEXT = 'Error: table does not exist.';
ELSE

    -- Retrieve passenger details for given route range

SELECT          customer_id, route_id, depart, arrival, seat_num
FROM            passengers_on_flights
WHERE           depart = start_route AND arrival = end_route;
END IF;
END //

DELIMITER ;

CALL get_passenger_details('NYX', 'NDU');
```

17. Write a query to create a stored procedure that extracts all the details from the routes table where the travelled distance is more than 2000 miles.

```
USE `air_cargo`;
DROP procedure IF EXISTS `dm`;


DELIMITER $$
USE `air_cargo`$$
CREATE PROCEDURE `dm` ()
BEGIN
SELECT      *
FROM        routes
WHERE       distance_miles > 2000
ORDER BY    route_id ASC;
END$$
DELIMITER ;

CALL dm();
```

route_id	flight_num	origin_airport	destination_airport	aircraft_id	distance_miles
1	1111	EWR	HNL	767-301ER	4962
2	1112	HNL	EWR	767-301ER	4962
3	1113	EWR	LHR	A321	3466
4	1114	JFK	LAX	767-301ER	2475
5	1115	LAX	JFK	767-301ER	2475
6	1116	HNL	LAX	767-301ER	2556
10	1120	HNL	DEN	A321	3365
12	1122	ABI	ADK	767-301ER	4300
13	1123	ADK	BQN	A321	2232
14	1124	BQN	CAK	A321	2445
18	1128	ANI	BGR	ERJ142	2450
19	1129	ATW	AVL	A321	2222
20	1130	AVL	BOI	767-301ER	3134
21	1131	BFL	BET	A321	2425
23	1133	BLV	BFL	767-301ER	2354
25	1135	RDM	BJI	A321	2425
34	1144	CRW	COD	A321	2452
35	1145	STT	CDB	ERJ142	2121
43	1153	CBM	BOI	A321	8989
44	1154	COU	CAK	767-301ER	7676
46	1156	CDV	HNL	767-301ER	8668
48	1158	SCC	DEN	A321	5645
49	1159	DEC	ABI	A321	4533
50	1160	DRT	ORD	A321	2445

18. Write a query to create a stored procedure that groups the distance travelled by each flight into three categories. The categories are, short distance travel (SDT) for >=0 AND <= 2000 miles, intermediate distance travel (IDT) for >2000 AND <=6500, and long-distance travel (LDT) for >6500.

```
USE `air_cargo`;
DROP procedure IF EXISTS `fc`;

DELIMITER $$
USE `air_cargo`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `fc`()
BEGIN
SELECT      flight_num, distance_miles,
CASE
WHEN        distance_miles > 6500 THEN 'long-distance travel (LDT)'
WHEN        distance_miles  2000 AND distance_miles <= 6500 THEN 'intermediate distance travel (IDT)'
ELSE        'short distance travel (SDT)'
END AS      flight_categories
FROM        routes;
END$$

DELIMITER ;

CALL        fc();
```

flight_num	distance_miles	flight_categories
1111	4962	intermediate distance travel (IDT)
1112	4962	intermediate distance travel (IDT)
1113	3466	intermediate distance travel (IDT)
1114	2475	intermediate distance travel (IDT)
1115	2475	intermediate distance travel (IDT)
1116	2556	intermediate distance travel (IDT)
1117	1745	short distance travel (SDT)
1118	719	short distance travel (SDT)
1119	862	short distance travel (SDT)
1120	3365	intermediate distance travel (IDT)
1122	4300	intermediate distance travel (IDT)
1123	2232	intermediate distance travel (IDT)
1124	2445	intermediate distance travel (IDT)
1125	2000	short distance travel (SDT)
1126	1700	short distance travel (SDT)
1127	1900	short distance travel (SDT)
1128	2450	intermediate distance travel (IDT)
1129	2222	intermediate distance travel (IDT)
1130	3134	intermediate distance travel (IDT)
1131	2425	intermediate distance travel (IDT)
1132	1242	short distance travel (SDT)
1133	2354	intermediate distance travel (IDT)
1134	1575	short distance travel (SDT)
1135	2425	intermediate distance travel (IDT)
1136	1311	short distance travel (SDT)
1137	578	short distance travel (SDT)
1138	246	short distance travel (SDT)
1139	909	short distance travel (SDT)
1140	780	short distance travel (SDT)
1141	660	short distance travel (SDT)
1142	246	short distance travel (SDT)
1143	1345	short distance travel (SDT)
1144	2452	intermediate distance travel (IDT)
1145	2121	intermediate distance travel (IDT)
1146	1212	short distance travel (SDT)
1147	999	short distance travel (SDT)
1148	1111	short distance travel (SDT)
1149	1579	short distance travel (SDT)
1150	909	short distance travel (SDT)
1151	898	short distance travel (SDT)
1152	890	short distance travel (SDT)
1153	8989	long-distance travel (LDT)
1154	7676	long-distance travel (LDT)
1155	676	short distance travel (SDT)
1156	8668	long-distance travel (LDT)
1157	675	short distance travel (SDT)
1158	5645	intermediate distance travel (IDT)
1159	4533	intermediate distance travel (IDT)
1160	2445	intermediate distance travel (IDT)

19. Write a query to extract ticket purchase date, customer ID, class ID and specify if the complimentary services are provided for the specific class using a stored function in stored procedure on the ticket_details table.
Condition: If the class is Business and Economy Plus, then complimentary services are given as Yes, else it is No

-- Creating a fn to check the complimentary service condition

```
USE `air_cargo`;
DROP function IF EXISTS `complimentary_service`;

DELIMITER $$
USE `air_cargo`$$
CREATE FUNCTION     `complimentary_service` (class_id VARCHAR(100))
RETURNS             VARCHAR(3)
READS SQL DATA
DETERMINISTIC
BEGIN
RETURN              if(class_id IN ('Bussiness', 'Economy Plus'), 'Yes', 'No');
END$$

DELIMITER ;
```

-- Creating a procedure with the function

```
USE `air_cargo`;
DROP procedure IF EXISTS `proc_compservices`;

DELIMITER $$
USE `air_cargo`$$
CREATE PROCEDURE `proc_compservices` ()
BEGIN
SELECT              customer_id, Class_id,
                    complimentary_service(class_id)
FROM                ticket_details
ORDER BY            customer_id ASC;
END$$

DELIMITER ;

CALL                proc_compservices();
```

customer_id	Class_id	complimentary_service(class_id)
1	First Class	No
1	Economy Plus	Yes
2	Economy	No
2	Bussiness	Yes
4	First Class	No
4	First Class	No
5	Economy	No
5	Bussiness	Yes
5	Economy	No
7	Bussiness	Yes
8	Economy Plus	Yes
8	Economy	No
9	First Class	No
9	First Class	No
10	Economy	No
11	Bussiness	Yes
11	Economy Plus	Yes
11	Bussiness	Yes
13	First Class	No
14	Economy	No
14	Economy	No
15	Bussiness	Yes
16	First Class	No
17	Economy Plus	Yes
18	Economy	No
18	First Class	No
19	Economy Plus	Yes
19	Economy	No
19	Economy Plus	Yes
20	First Class	No
20	First Class	No
21	Bussiness	Yes
22	Economy Plus	Yes
24	Bussiness	Yes
25	Economy	No
25	Bussiness	Yes
27	Economy	No
28	Economy	No
29	Bussiness	Yes
29	Bussiness	Yes
31	Economy	No
32	Economy Plus	Yes
33	Bussiness	Yes
41	First Class	No
44	First Class	No
46	Economy	No
46	First Class	No
47	Economy Plus	Yes
49	Bussiness	Yes
50	Economy Plus	Yes

20. Write a query to extract the first record of the customer whose last name ends with Scott using a cursor from the customer table.

```
USE `air_cargo`;
DROP procedure IF EXISTS `Proc_lnscott`;

DELIMITER $$
USE `air_cargo`$$
CREATE PROCEDURE  `Proc_lnscott` ()
BEGIN DECLARE     c_id int;
DECLARE           f_name VARCHAR(100);
DECLARE           l_name VARCHAR(100);
DECLARE           dob date;
DECLARE           sex VARCHAR(10);
DECLARE           cursor_1 CURSOR FOR
                                SELECT      customer_id, first_name,
                                            last_name, date_of_birth, gender
                                FROM        customer
                                WHERE       last_name = 'Scott'
                                Limit       1;
OPEN              cursor_1;
REPEAT FETCH      cursor_1 INTO c_id,f_name,l_name, dob, sex;
UNTIL             sex = 'M' OR sex = 'F' END REPEAT;
SELECT            c_id as customer_id, f_name as first_name, l_name as last_name,
                  dob as date_of_birth, sex as gender;
CLOSE             cursor_1;
END$$

DELIMITER ;

CALL              Proc_lnscott();
```

customer_id	first_name	last_name	date_of_birth	gender
37	Samuel	Scott	2000-01-28	M