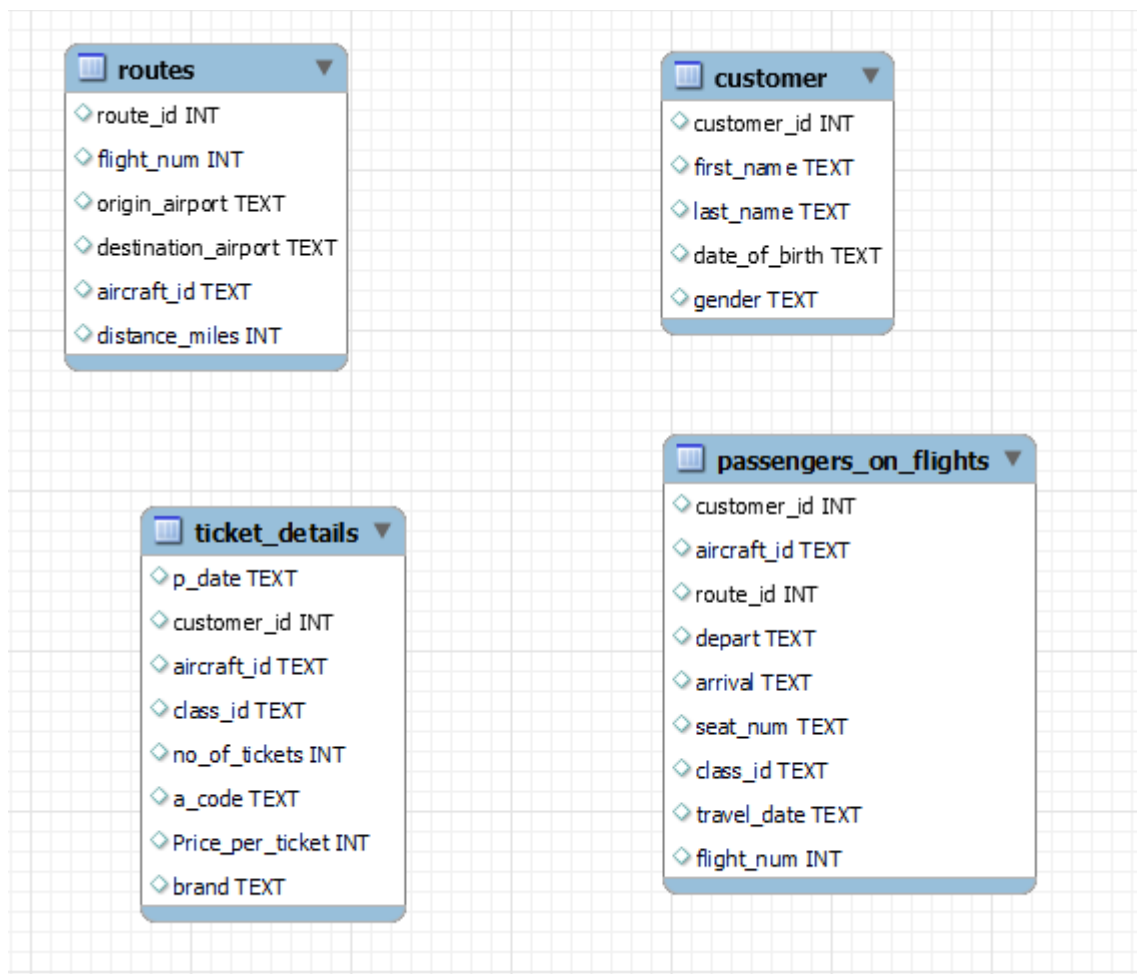


# Course-End Project

## Air Cargo Analysis

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.

```
1 CREATE TABLE route_details (  
2     route_id INT NOT NULL UNIQUE,  
3     flight_num INT NOT NULL CHECK (flight_num > 0),  
4     origin_airport CHAR (3) NOT NULL,  
5     destination_airport CHAR (3) NOT NULL,  
6     aircraft_id INT NOT NULL,  
7     distance_miles SMALLINT NOT NULL CHECK (distance_miles > 0),  
8     PRIMARY KEY (route_id)  
9 );  
10  
11  
12
```

Output



Action Output

#	Time	Action	Message
✓ 1	23:01:21	CREATE TABLE route_details ( route_id INT NOT NULL UNIQUE, flight_num INT NOT NULL...	0 row(s) affected

### 3. Write a query to display all the passengers who have travelled in routes 01 to 25 from the passengers\_on\_flights table



```
1 • SELECT * FROM passengers_on_flights where route_id between 01 and 25;
```

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

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

```
1 SELECT COUNT(class_id) Bussiness_Class, SUM(price_per_ticket) revenue_in_bussiness FROM ticket_details WHERE class_id = 'Bussiness';
```

<

Result Grid  Filter Rows:  Export:  Wrap Cell Content: 

	Bussiness_Class	revenue_in_bussiness
▶	13	6034

Output

 Action Output ▼

#	Time	Action	Message
✓ 1	21:23:48	SELECT COUNT(class_id) Bussiness_Class, SUM(price_per_ticket) revenue_in_bussiness FR...	1 row(s) returned

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

```
1  SELECT CONCAT(first_name, ' ', last_name) FULL_NAME FROM customer;
2
3
```

Result Grid

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

Result 5 x

Output

Action Output

#	Time	Action	Message
✓ 1	22:20:46	SELECT CONCAT(first_name, ' ', last_name) FULL_NAME FROM customer	50 row(s) returned

**6. Write a query to extract the customers who have registered and booked a ticket from the customer and ticket\_details tables.**

```

1 • SELECT customer.customer_id, customer.first_name, customer.last_name, ticket_details.p_date
2 FROM customer
3 INNER JOIN ticket_details
4 ON customer.customer_id = ticket_details.customer_id;
5
6
7

```



Result Grid				
Filter Rows:		Export:		
Wrap Cell Content:				
	customer_id	first_name	last_name	p_date
▶	27	Cherly	Vernon	26-12-2018
	22	Pheny	Eri	02-02-2020
	21	Chirsty	Josh	03-03-2020
	4	Cathenna	Emily	04-04-2020
	5	Aaron	Kim	05-05-2020
	7	Anderson	Stewart	07-07-2020
	8	Floyd	Ted	08-08-2020
	9	Leo	Travis	09-09-2020
	10	Melvin	Tracy	10-10-2020
	11	Roger	Walson	11-11-2020
	19	Joyce	Paul	12-12-2020
	13	Solomon	Walter	01-01-2019
	14	Carol	Vernon	02-02-2019
	25	Moss	Morris	03-03-2019
	16	Chirstine	Willis	04-04-2019
	17	Catherine	Shad	03-05-2019
	18	Gloria	Richie	06-06-2019
	24	Calvin	Willis	07-07-2019

**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.**

```

1 • SELECT first_name, last_name FROM customer
2 WHERE
3     customer_id IN (SELECT customer_id FROM ticket_details WHERE brand = 'Emirates');
4
5
6

```

<		
Result Grid		
Filter Rows: <input type="text"/>		
Export:  Wrap Cell Content: 		
	first_name	last_name
▶	Steve	Ryan
	Cathenna	Emily
	Aaron	Kim
	Anderson	Stewart
	Leo	Travis
	Roger	Walson
	Carol	Vernon
	Gloria	Richie
	Joyce	Paul
	Moss	Morris
	Cherly	Vernon
	James	Robert
	Bily	Brian
	Russell	Peter

**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.**

```

1 • SELECT customer_id, aircraft_id, route_id, depart, arrival, seat_num, class_id, travel_date, flight_num FROM passengers_on_flights
2 GROUP BY customer_id, aircraft_id, route_id, depart, arrival, seat_num, class_id, travel_date, flight_num
3 HAVING class_id = 'Economy Plus';
4
5

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	customer_id	aircraft_id	route_id	depart	arrival	seat_num	class_id	travel_date	flight_num
▶	1	ERJ142	9	DEN	LAX	01EP	Economy Plus	26-12-2019	1119
	8	A321	38	CST	DAL	02EP	Economy Plus	09-08-2020	1148
	11	ERJ142	31	BTM	CHA	03EP	Economy Plus	02-08-2018	1141
	17	A321	13	ABI	ADK	04EP	Economy Plus	03-06-2019	1123
	19	CRJ900	47	DAL	LAX	05EP	Economy Plus	13-01-2021	1157
	19	CRJ900	30	BUR	STT	06EP	Economy Plus	17-12-2020	1140
	22	ERJ142	22	BGR	BJI	07EP	Economy Plus	09-02-2020	1132
	32	ERJ142	31	BTM	CHA	08EP	Economy Plus	04-03-2021	1141
	47	CRJ900	33	CDC	CST	09EP	Economy Plus	15-12-2020	1143
	50	A321	21	BFL	BET	10EP	Economy Plus	15-08-2020	1131

passengers\_on\_flights 10 x

Output

Action Output

#	Time	Action	Message
✓ 1	22:56:56	SELECT customer_id, aircraft_id, route_id, depart, arrival, seat_num, class_id, travel_date, flig...	10 row(s) returned

**9. Write a query to identify whether the revenue has crossed 10000 using the IF clause on the ticket\_details table.**



```
1 • SELECT IF(SUM(Price_per_ticket)>10000,'Revenue is greater than 10000','Revenue is less than 10000')
2   Check_if_revenu_greater_than_10000 from ticket_details
3
4
5
6
```

< Result Grid Filter Rows:  | Export: | Wrap Cell Content:

Check_if_revenu_greater_than_10000
▶ Revenue is greater than 10000

Result 11 ×

Output

Action Output

#	Time	Action	Message
✓ 1	23:08:19	SELECT IF(SUM(Price_per_ticket)>10000,'Revenue is greater than 10000','Revenue is less th...	1 row(s) returned

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

1 • CREATE USER junior@localhost

2 IDENTIFIED BY 'junior123';

3

4

Output

Action Output

#	Time	Action	Message
✓ 1	23:12:21	CREATE USER junior@localhost IDENTIFIED BY 'junior123'	0 row(s) affected

1 • GRANT EXECUTE ON securitydb.\*

2 TO junior@localhost;

3

4

Output

Action Output

#	Time	Action
✓ 1	23:15:40	GRANT EXECUTE ON securitydb.* TO junior@localhost

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

1 • SELECT \*, MAX(Price\_per\_ticket) OVER ( PARTITION BY class\_id ) max\_price  
2 FROM ticket\_details;  
3  
4  
5

Result Grid

Filter Rows:




Export:

Wrap Cell Content:

	p_date	customer_id	aircraft_id	class_id	no_of_tickets	a_code	Price_per_ticket	brand	max_price
▶	03-03-2019	25	767-301ER	Bussiness	1	BHX	499	Emirates	510
	17-07-2020	49	767-301ER	Bussiness	1	BFS	430	Emirates	510
	03-03-2020	21	CRJ900	Bussiness	1	BOH	490	Bristish Airways	510
	12-03-2020	33	CRJ900	Bussiness	1	BOH	490	Bristish Airways	510
	01-04-2018	29	ERJ142	Bussiness	1	EME	510	Jet Airways	510
	07-07-2020	7	767-301ER	Bussiness	1	BFS	430	Emirates	510
	07-07-2019	24	A321	Bussiness	1	CTM	480	Qatar Airways	510
	01-11-2018	15	A321	Bussiness	1	BFS	430	Qatar Airways	510
	25-01-2019	2	A321	Bussiness	1	YVR	505	Qatar Airways	510
	11-11-2020	11	767-301ER	Bussiness	1	AGB	465	Emirates	510
	22-10-2019	29	A321	Bussiness	1	PEK	410	Qatar Airways	510
	01-07-2018	5	767-301ER	Bussiness	1	BFS	430	Emirates	510
	08-11-2020	11	767-301ER	Bussiness	1	AGB	465	Emirates	510
	26-12-2018	27	767-301ER	Economy	1	DAL	130	Emirates	190
	01-09-2018	2	767-301ER	Economy	1	DAL	130	Emirates	190
	01-12-2018	28	ERJ142	Economy	1	BHX	170	Jet Airways	190
	05-05-2020	5	ERJ142	Economy	1	CTM	120	Jet Airways	190
	02-02-2019	14	ERJ142	Economy	1	CTM	120	Jet Airways	190
	19-12-2018	31	767-301ER	Economy	1	DAL	130	Emirates	190
	10-10-2020	10	A321	Economy	1	MCO	135	Qatar Airways	190
	30-05-2020	5	ERJ142	Economy	1	CTM	120	Jet Airways	190

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.

```
1 • CREATE INDEX route_id ON passengers_on_flights (route_id);
2
3 • SELECT customer_id,depart,arrival,aircraft_id FROM passengers_on_flights WHERE route_id = '04';
4
5
```

<				
Result Grid    Filter Rows: <input type="text"/>   Export:    Wrap Cell Content: 				
	customer_id	depart	arrival	aircraft_id
▶	2	JFK	LAX	767-301ER
	4	JFK	LAX	767-301ER
	11	JFK	LAX	767-301ER

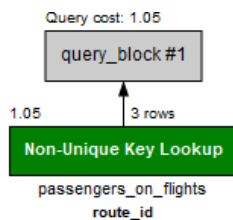
**13. For the route ID 4, write a query to view the execution plan of the passengers\_on\_flights table.**

```

1 • SELECT * FROM passengers_on_flights WHERE route_id = '04';
2
3
4
5

```

	customer_id	aircraft_id	route_id	depart	arrival	seat_num	class_id	travel_date	flight_num
▶	2	767-301ER	4	JFK	LAX	01E	Economy	02-09-2018	1114
	4	767-301ER	4	JFK	LAX	03FC	First Class	30-04-2020	1114
	11	767-301ER	4	JFK	LAX	05B	Bussiness	09-11-2020	1114





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

```

1 • SELECT customer_id, aircraft_id, SUM(Price_per_ticket) as Total_Price
2 FROM ticket_details
3 GROUP BY customer_id, aircraft_id WITH ROLLUP;
4
5
6
7
8

```

Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
	customer_id	aircraft_id	Total_Price
▶	1	CRJ900	320
	1	ERJ142	250
	1	NULL	570
	2	767-301ER	130
	2	A321	505
	2	NULL	635
	4	767-301ER	780
	4	NULL	780
	5	767-301ER	430
	5	ERJ142	240
	5	NULL	670
	7	767-301ER	430

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

```

1 • CREATE VIEW BussinessClassView AS SELECT customer_id, class_id, brand from ticket_details where class_id = "Bussiness";
2
3 • select * from BussinessClassView;
4

```

Result Grid			
Filter Rows:			
Export: Wrap Cell Content:			
	customer_id	class_id	brand
▶	21	Bussiness	Bristish Airways
	7	Bussiness	Emirates
	11	Bussiness	Emirates
	25	Bussiness	Emirates
	24	Bussiness	Qatar Airways
	29	Bussiness	Qatar Airways
	2	Bussiness	Qatar Airways
	29	Bussiness	Jet Airways
	5	Bussiness	Emirates
	15	Bussiness	Qatar Airways
	33	Bussiness	Bristish Airways
	49	Bussiness	Emirates
	11	Bussiness	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.**

```

1 • DROP PROCEDURE IF EXISTS getPassengersBetweenRoutes;
2 DELIMITER &&
3 • CREATE PROCEDURE getPassengersBetweenRoutes(
4     IN route_a INT,
5     IN route_b INT)
6 BEGIN
7     DECLARE EXIT HANDLER FOR SQLEXCEPTION
8     BEGIN
9         GET DIAGNOSTICS CONDITION 1
10        @sqlstate = RETURNED_SQLSTATE,
11        @errno = MYSQL_ERRNO,
12        @text = MESSAGE_TEXT;
13        SET @full_error = CONCAT("SQLException Handler - ERROR ", @errno, " (", @sqlstate, "): ", @text);
14        SELECT @full_error AS msg;
15    END;
16    -- MySQL Query
17    SELECT * FROM passengers_on_flights WHERE route_id BETWEEN route_a AND route_b;
18 END &&
19 • CALL getPassengersBetweenRoutes(1,25);

```

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

**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.**



```

1 DELIMITER &&
2 • CREATE PROCEDURE Get_Customers_With_2K()
3 BEGIN
4 SELECT * FROM routes where distance_miles>2000;
5 END &&
6 • CALL Get_Customers_With_2K();
7

```

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

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  $\geq 0$  AND  $\leq 2000$  miles, intermediate distance travel (IDT) for  $>2000$  AND  $\leq 6500$ , and long-distance travel (LDT) for  $>6500$ .

```

1 • DROP PROCEDURE IF EXISTS getTravelType;
2
3 DELIMITER $$
4 • CREATE PROCEDURE getTravelType()
5 BEGIN
6     DECLARE dist INT DEFAULT 0;
7     DECLARE done BIT DEFAULT FALSE;
8     DECLARE route_cur CURSOR FOR SELECT distance_miles FROM routes;
9     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
10    OPEN route_cur;
11
12    CREATE TEMPORARY TABLE IF NOT EXISTS routes_2 (
13        travel_type VARCHAR(3)
14    ) AS ( SELECT * FROM routes);
15
16    while_label: WHILE done = FALSE DO
17        FETCH route_cur INTO dist;
18        IF dist >=0 AND dist <= 2000 THEN
19            UPDATE routes 2
20                SET travel_type = "SDT"
21                WHERE distance_miles = dist;
22
23            ELSEIF dist >2000 AND dist <=6500 THEN
24                UPDATE routes_2
25                SET travel_type = "IDT"
26                WHERE distance_miles = dist;
27
28            ELSEIF dist >6500 THEN
29                UPDATE routes_2
30                SET travel_type = "LDT"
31                WHERE distance_miles = dist;
32                .
33            ELSE
34                ITERATE while_label;
35            END IF;
36        END WHILE while_label;
37
38    SELECT * FROM routes 2 ORDER BY distance miles;

```

39    END\$\$

40

41    CALL getTravelType();

42

Result Grid							
		Filter Rows:			Export:	Wrap Cell Content:	
	travel_type	route_id	flight_num	origin_airport	destination_airport	aircraft_id	distance_miles
▶	SDT	28	1138	BOS	CDC	767-301ER	246
	SDT	32	1142	CLD	CHI	767-301ER	246
	SDT	27	1137	BOI	CLD	A321	578
	SDT	31	1141	BTM	CHA	ERJ142	660
	SDT	47	1157	DAL	LAX	CRJ900	675
	SDT	45	1155	CCR	EWR	CRJ900	676
	SDT	8	1118	ORD	EWR	A321	719
	SDT	30	1140	BUR	STT	CRJ900	780
	SDT	9	1119	DEN	LAX	ERJ142	862
	SDT	42	1152	CSG	BOS	767-301ER	890
	SDT	41	1151	CAE	DRT	ERJ142	898
	SDT	29	1139	BKG	CRW	767-301ER	909
	SDT	40	1150	CDB	DEC	A321	909
	SDT	37	1147	CHI	CST	767-301ER	999
	SDT	38	1148	CST	DAL	A321	1111
	SDT	36	1146	CHA	COU	CRJ900	1212
	SDT	22	1132	BGR	BJI	ERJ142	1242
	SDT	26	1136	BET	BTM	ERJ142	1311
	SDT	33	1143	CDC	CST	CRJ900	1345
	SDT	24	1134	BJI	BQN	A321	1575
	SDT	39	1149	COD	SCC	CRJ900	1579
	SDT	16	1126	ALB	APN	A321	1700

Result 1 ×

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*

Create stored function:

```
1  DELIMITER $$
2  • CREATE FUNCTION complimentary_servicess(class_id VARCHAR(225))
3  RETURNS VARCHAR(225) DETERMINISTIC
4  BEGIN DECLARE complimentary_servicess VARCHAR(225);
5  IF class_id = 'Bussiness' THEN SET complimentary_servicess = 'Yes';
6  ELSEIF class_id = 'Economy Plus' THEN SET complimentary_servicess = 'Yes';
7  ELSE SET complimentary_servicess = 'No' ;
8  END IF;
9  RETURN (complimentary_servicess);
10 END$$
11 DELIMITER $$;
```

Create stored procedure to call stored function:

```
12 DELIMITER $$
13 • CREATE PROCEDURE Extra_complimentary_services()
14 BEGIN
15 SELECT p_date, customer_id, class_id, complimentary_servicess(class_id) as Extra_complimentary_services
16 FROM ticket_details
17 ORDER BY class_id;
18 END$$ DELIMITER $$ ;
19 call Extra_complimentary_services();
20
21
```

	p_date	customer_id	class_id	Extra_complimentary_services
	11-11-2020	11	Bussiness	Yes
	22-10-2019	29	Bussiness	Yes
	01-07-2018	5	Bussiness	Yes
	08-11-2020	11	Bussiness	Yes
	26-12-2018	27	Economy	No
	01-09-2018	2	Economy	No
	01-12-2018	28	Economy	No
	05-05-2020	5	Economy	No
	02-02-2019	14	Economy	No
	19-12-2018	31	Economy	No
	10-10-2020	10	Economy	No
	30-05-2020	5	Economy	No
	07-10-2020	46	Economy	No
	06-06-2019	18	Economy	No
	21-09-2019	25	Economy	No
	24-12-2019	14	Economy	No
	01-02-2018	19	Economy	No
	01-05-2018	8	Economy	No
	02-02-2020	22	Economy...	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.

```
1  DELIMITER $$
2  • CREATE PROCEDURE first_last_name_is_scott()
3  BEGIN DECLARE a VARCHAR(255);
4  DECLARE b VARCHAR(255);
5  DECLARE cursor_1 CURSOR FOR SELECT first_name,last_name FROM customer
6  WHERE last_name = 'Scott';
7  OPEN cursor_1;
8  REPEAT FETCH cursor_1 INTO a,b;
9  UNTIL b = 0 END REPEAT;
10 SELECT a as first_name, b as last_name;
11 CLOSE cursor_1;
12 END;
13 $$ DELIMITER ;
14 • call first_last_name_is_scott();
15
```

<		
Result Grid		
Filter Rows:		
Export:		
Wrap Cell Content:		
	first_name	last_name
▶	Samuel	Scott