SQL PROJECT-2

Air Cargo Analysis

.sql file: air_cargo_analysis
SQL Project 2 SAUM

SUBMITTED BY:

SAUMYA GUREJA

Air Cargo Analysis

Course-end Project 2

Description

Air Cargo is an aviation company that provides air transportation services for passengers and freight. Air Cargo uses its aircraft to provide different services with the help of partnerships or alliances with other airlines. The company wants to prepare reports on regular passengers, busiest routes, ticket sales details, and other scenarios to improve the ease of travel and booking for customers.

Project Objective:

You, as a DBA expert, need to focus on identifying the regular customers to provide offers, analyze the busiest route which helps to increase the number of aircraft required and prepare an analysis to determine the ticket sales details. This will ensure that the company improves its operability and becomes more customer-centric and a favorable choice for air travel.

Note: You must download the dataset from the course resource section in the LMS and create the tables to perform the above objective.

Dataset description:

customer: Contains the information of customers

- customer_id – ID of the customer
- first_name – First name of the customer
- last_name – Last name of the customer
- date_of_birth – Date of birth of the customer
- gender – Gender of the customer

passengers_on_flights: Contains information about the travel details

- aircraft_id – ID of each aircraft in a brand
- route_id – Route ID of from and to location
- customer_id – ID of the customer
- depart – Departure place from the airport
- arrival – Arrival place in the airport

- seat_num – Unique seat number for each passenger
- class_id – ID of travel class
- travel_date – Travel date of each passenger
- flight_num – Specific flight number for each route

ticket_details: Contains information about the ticket details

- p_date – Ticket purchase date
- customer_id – ID of the customer
- aircraft_id – ID of each aircraft in a brand
- class_id – ID of travel class
- no_of_tickets – Number of tickets purchased
- a_code – Code of each airport
- price_per_ticket – Price of a ticket
- brand – Aviation service provider for each aircraft

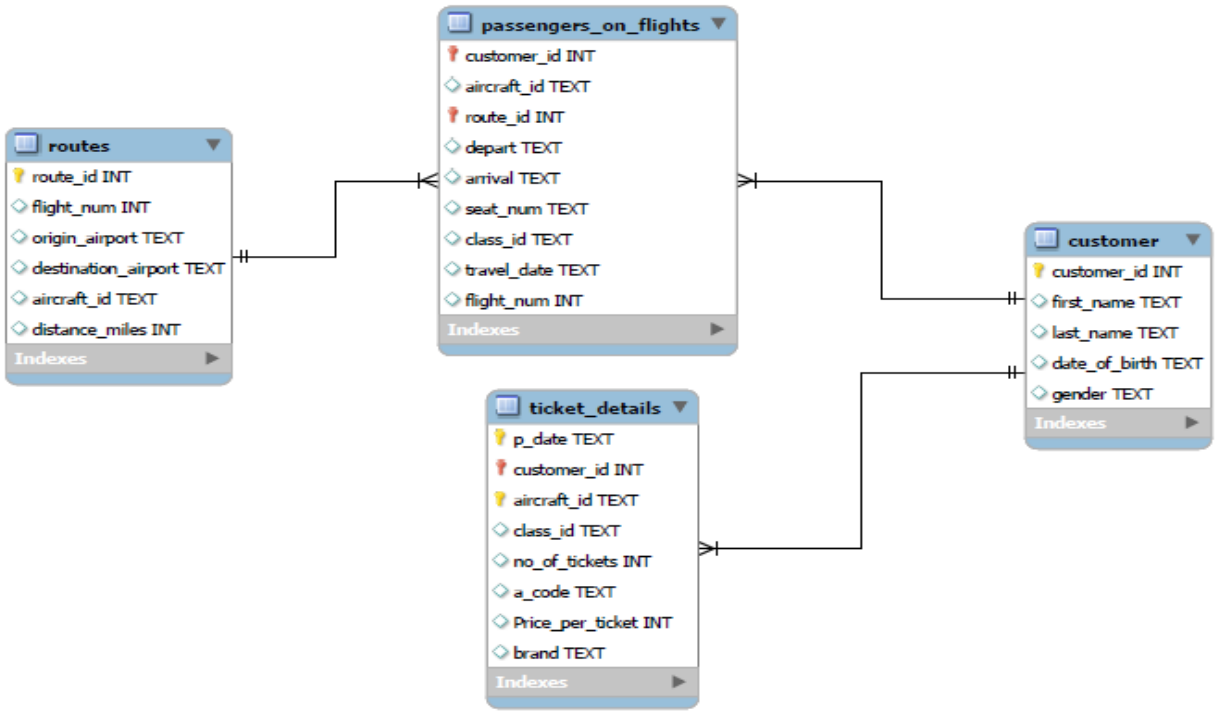
routes: Contains information about the route details

- Route_id – Route ID of from and to location
- Flight_num – Specific fight number for each route
- Origin_airport – Departure location
- Destination_airport – Arrival location
- Aircraft_id – ID of each aircraft in a brand
- Distance_miles – Distance between departure and arrival location

Datasets: air_cargo_datasets.zip

air_cargo_datasets.zip

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



er diagram air cargo_pdf_.pdf

Pdf link: [er diagram air cargo_pdf_.pdf]

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 UNIQUE, flight_num int CHECK(flight_num<>0), origin_airport varchar(200), destination_airport varchar(200), aircraft_id varchar(200), distance_miles int CHECK(distance_miles>0));

DESCRIBE route_details;

```
24 •  describe route_details;
```

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

Field	Type	Null	Key	Default	Extra
route_id	int	YES	UNI	NULL	
flight_num	int	YES		NULL	
origin_airport	varchar(50)	YES		NULL	
destination_airport	varchar(50)	YES		NULL	
aircraft_id	varchar(20)	YES		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 route_id, c.* FROM customer c LEFT JOIN passengers_on_flights pf ON c.customer_id=pf.customer_id WHERE route_id BETWEEN 01 AND 25 ORDER BY route_id asc;

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

route_id	customer_id	first_name	last_name	date_of_birth	gender
1	18	Gloria	Richie	04-12-1989	F
4	2	Steve	Ryan	03-04-1983	M
4	4	Cathenna	Emily	14-09-1977	F
4	11	Roger	Walson	24-05-1996	M
5	4	Cathenna	Emily	14-09-1977	F
5	11	Roger	Walson	24-05-1996	M
8	46	Louis	Douglas	22-09-1997	M
9	1	Julie	Sam	12-01-1989	F
9	29	Watson	Ronald	11-01-1991	M
10	10	Melvin	Tracy	23-04-1995	M
12	5	Aaron	Kim	18-02-1991	M
13	13	Solomon	Walter	26-07-1998	M
13	17	Catherine	Shad	09-11-1988	F

Result 4

Output

Action Output

#	Time	Action	Message
7	16:29:12	describe route_details	6 row(s) returned
8	16:37:50	select route_id, c.* from customer c left join passengers_on_flights pf on c.customer_id=p...	26 row(s) returned

---

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

SELECT sum(no_of_tickets) Total_Passengers, sum(no_of_tickets * Price_per_ticket) Revenue FROM ticket_details WHERE class_id="Bussiness";

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

Total_Passengers	Revenue
13	6034

Result 9

Output

Action Output

#	Time	Action	Message
12	16:51:16	SELECT sum(no_of_tickets) Total_Passengers, sum(no_of_tickets * Price_per_ticket) Re...	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.

SELECT concat(first_name," ",last_name) Full_Name FROM customer;

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

Full_Name
Julie Sam
Steve Ryan
Morris Lois
Cathenna Emily
Aaron Kim
Alexander Scot
Anderson Stewart
Floyd Ted
Leo Travis
Melvin Tracy
Roger Walson

Result 7

Output

Action Output

#	Time	Action	Message
10	16:44:02	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. Use data from the customer and ticket_details tables.

SELECT DISTINCT c.* FROM customer c Right Join ticket_details td ON c.customer_id = td.customer_id ORDER BY customer_id asc;

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

customer_id	first_name	last_name	date_of_birth	gender
1	Julie	Sam	12-01-1989	F
2	Steve	Ryan	03-04-1983	M
4	Cathenna	Emily	14-09-1977	F
5	Aaron	Kim	18-02-1991	M
7	Anderson	Stewart	11-01-1992	M
8	Floyd	Ted	21-02-1993	M
9	Leo	Travis	22-03-1994	M
10	Melvin	Tracy	23-04-1995	M
11	Roger	Walson	24-05-1996	M
13	Solomon	Walter	26-07-1998	M

Result 11

Output

Action Output

#	Time	Action	Message
14	16:55:12	SELECT DISTINCT c.* FROM customer c Right Join ticket_details td ON c.customer_id=t...	33 row(s) returned

---

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 c.customer_id, c.first_name, c.last_name FROM ticket_details td LEFT JOIN customer c ON td.customer_id = c.customer_id WHERE brand ="Emirates" ORDER BY customer_id asc;

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

customer_id	first_name	last_name
2	Steve	Ryan
4	Cathenna	Emily
5	Aaron	Kim
7	Anderson	Stewart
9	Leo	Travis
11	Roger	Walson
14	Carol	Vernon
18	Gloria	Richie
19	Joyce	Paul
25	Moss	Morris

Result 13

Output

Action Output

#	Time	Action	Message
16	17:00:07	SELECT DISTINCT c.customer_id,c.first_name, c.last_name FROM ticket_details td Left ...	14 row(s) returned

---

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 class_id, count(*) EcoPlus_Passengers FROM passengers_on_flights GROUP BY class_id HAVING class_id = "Economy Plus";

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

class_id	EcoPlus_Passengers
Economy Plus	10

Result 18

Output

Action Output

#	Time	Action	Message
28	17:08:48	SELECT class_id, count(*) EcoPlus_Passengers FROM passengers_on_flights GROUP B...	1 row(s) returned

---

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

SELECT sum(Price_per_ticket * no_of_tickets) Revenue, IF(sum(no_of_tickets * Price_per_ticket)>10000, "Target Achieved","Target Not Achieved") Status FROM ticket_details;

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

Revenue	Status
15369	Target Achieved

Result 24

Output

Action Output

#	Time	Action	Message
34	17:15:10	SELECT sum(Price_per_ticket * no_of_tickets) Revenue, if(sum(no_of_tickets*Price_per_t...	1 row(s) returned

---

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

CREATE USER "data_analyst"@"localhost" IDENTIFIED BY "data_analyst";

SELECT user, host FROM mysql.user WHERE user="data_analyst";

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

user	host
data_analyst	localhost

user 26

Output

Action Output

#	Time	Action	Message
37	18:22:49	SELECT user, host FROM mysql.user WHERE user="data_analyst"	1 row(s) returned

GRANT select, insert ON air_cargo_project.* TO "data_analyst"@"localhost";

SELECT * FROM mysql.db WHERE user="data_analyst" and Db="air_cargo_project";

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Host	Db	User	Select_priv	Insert_priv	Update_priv	Delete_priv	Create_priv	Drop_priv	Grant_priv	References_p
localhost	air_cargo_project	data_analyst	Y	Y	N	N	N	N	N	N
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

db 30

Output

Action Output

#	Time	Action	Message
46	18:38:31	SELECT * FROM mysql.db WHERE user="data_analyst" and Db="air_cargo_project"	1 row(s) returned

---

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 ORDER BY class_id asc) Max_Ticket_Price FROM ticket_details;

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

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

Result 35

Output

Action Output

#	Time	Action	Message
51	18:56:49	SELECT DISTINCT class_id, MAX(Price_per_ticket) OVER (PARTITION BY class_id OR...	4 row(s) returned

---

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.

ALTER TABLE passengers_on_flights ADD PRIMARY KEY (route_id, customer_id);

SHOW INDEX FROM passengers_on_flights WHERE Key_name = 'PRIMARY';

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

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
passengers_on_flights	0	PRIMARY	1	route_id	A	32	NULL	NULL		BTREE
passengers_on_flights	0	PRIMARY	2	customer_id	A	50	NULL	NULL		BTREE

Result 38

Output

Action Output

#	Time	Action	Message
57	19:26:20	SHOW INDEX FROM passengers_on_flights WHERE Key_name = 'PRIMARY'	2 row(s) returned

SELECT customer_id FROM passengers_on_flights WHERE route_id=4;

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

customer_id
2
4
11

passengers_on_flights 41

Output

Action Output

#	Time	Action	Message
61	19:32:07	SELECT customer_id FROM passengers_on_flights WHERE route_id = 4	3 row(s) returned

---

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

EXPLAIN SELECT customer_id FROM passengers_on_flights WHERE route_id = 4;

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

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	passengers_on_flights	NULL	ref	PRIMARY	PRIMARY	4	const	3	100.00	Using index

Result 48

Output

Action Output

#	Time	Action	Message
74	19:58:40	EXPLAIN SELECT customer_id FROM passengers_on_flights WHERE route_id = 4	1 row(s) returned

---

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

SELECT td.customer_id, COALESCE(td.aircraft_id, 'Total Spent by this customer') AS Aircraft, SUM(td.no_of_tickets * td.Price_per_ticket) Total_Ticket_Price FROM ticket_details td GROUP BY td.customer_id, td.aircraft_id WITH ROLLUP;

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

customer_id	Aircraft	Total_Ticket_Price
1	CRJ900	320
1	ERJ142	250
1	Total Spent by this customer	570
2	767-301ER	130
2	A321	505
2	Total Spent by this customer	635
4	767-301ER	780
4	Total Spent by this customer	780
5	767-301ER	430
5	ERJ142	240
5	Total Spent by this customer	670
7	767-301ER	430
7	Total Spent by this customer	430

Result 13

Output

Action Output

#	Time	Action	Message
14	20:29:35	SELECT td.customer_id, COALESCE(td.aircraft_id, 'Total Spent by this customer') A...	75 row(s) returned

---

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

```
CREATE  VIEW business_class_customers AS

SELECT t.customer_id, c.first_name,c.last_name,t.brand FROM ticket_details t, customer c

WHERE t.customer_id = c.customer_id and t.class_id = 'Bussiness';


SELECT * FROM business_class_customers;
```

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

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

business_class_customers 25

Output

Action Output

#	Time	Action	Message
44	21:18:10	select * from business_class_customers	13 row(s) returned

---

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.

```
DELIMITER $$

CREATE PROCEDURE passengers_between_route_range(IN start_route_id INT, IN end_route_id INT)

BEGIN

  IF NOT EXISTS (SELECT * FROM information_schema.tables

          WHERE table_name = 'passengers_on_flights' AND table_schema = DATABASE()) THEN

    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error: Table passengers_on_flights does not exist.';

  ELSE

    SELECT pf.customer_id, c.first_name, c.last_name, pf.route_id, pf.depart, pf.arrival

    FROM passengers_on_flights pf

    JOIN customer c ON pf.customer_id = c.customer_id

    WHERE pf.route_id BETWEEN start_route_id AND end_route_id;

  END IF;

END$$

DELIMITER ;


CALL passengers_between_route_range(1,10);
```

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

customer_id	first_name	last_name	route_id	depart	arrival
1	Julie	Sam	9	DEN	LAX
2	Steve	Ryan	4	JFK	LAX
4	Cathenna	Emily	5	LAX	JFX
4	Cathenna	Emily	4	JFK	LAX
10	Melvin	Tracy	10	HNL	DEN

Result 11

Output

Action Output

#	Time	Action	Message
49	22:07:03	CALL passengers_between_route_range(1,10)	10 row(s) returned

---

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.

```
DELIMITER //

CREATE PROCEDURE long_distance_route()

BEGIN

  IF NOT EXISTS (SELECT * FROM information_schema.tables

          WHERE table_name = 'routes'

          AND table_schema = DATABASE()) THEN

    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error: Table routes does not exist.';

  ELSE

    SELECT * FROM routes

    WHERE distance_miles > 2000;

  END IF;

END //


CALL long_distance_route();
```

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

route_id	flight_num	origin_airport	destination_airport	aircraft_id	distance_miles
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

Result 26

Output

Action Output

#	Time	Action	Message
57	22:18:06	CALL long_distance_route(); DROP PROCEDURE IF EXISTS long_distance_routes(); --...	24 row(s) returned

---

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.

```
DELIMITER //
CREATE PROCEDURE flight_distance_category()
BEGIN
  IF NOT EXISTS (SELECT * FROM information_schema.tables
          WHERE table_name = 'routes'
          AND table_schema = DATABASE()) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error: Table routes does not exist.';
  ELSE
    SELECT
      flight_num,
      CASE
        WHEN distance_miles >= 0 AND distance_miles <= 2000 THEN 'Short Distance Travel'
        WHEN distance_miles > 2000 AND distance_miles <= 6500 THEN 'Intermediate Distance Travel'
        WHEN distance_miles > 6500 THEN 'Long Distance Travel'
      END AS distance_category,
      distance_miles
    FROM routes;
  END IF;
END //
DELIMITER ;

CALL flight_distance_category();
```

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

flight_num	distance_category	distance_miles
1111	Intermediate Distance Travel	4962
1112	Intermediate Distance Travel	4962
1113	Intermediate Distance Travel	3466
1114	Intermediate Distance Travel	2475
1115	Intermediate Distance Travel	2475
1116	Intermediate Distance Travel	2556
1117	Short Distance Travel	1745
1118	Short Distance Travel	719
1119	Short Distance Travel	862
1120	Intermediate Distance Travel	3365
1122	Intermediate Distance Travel	4300
1123	Intermediate Distance Travel	2232
1124	Intermediate Distance Travel	2445

Result 31

Output

Action Output

#	Time	Action	Message
63	22:27:51	CALL flight_distance_category()	49 row(s) returned

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*.

```
DELIMITER //
CREATE FUNCTION complimentary_services(class_id VARCHAR(50))
RETURNS VARCHAR(3)
DETERMINISTIC
BEGIN
  DECLARE service VARCHAR(3);
  IF class_id IN ('Business', 'Economy Plus') THEN
    SET service = 'Yes';
  ELSE
    SET service = 'No';
  END IF;
  RETURN service;
END //
DELIMITER ;

DELIMITER //
CREATE PROCEDURE ticket_details_with_services()
BEGIN
  IF NOT EXISTS (SELECT * FROM information_schema.tables
          WHERE table_name = 'ticket_details'
          AND table_schema = DATABASE()) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error: Table ticket_details does not exist.';
  ELSE
    SELECT
      p_date AS ticket_purchase_date,
      customer_id,
      class_id,
      complimentary_services(class_id)
    FROM ticket_details;
  END IF;
END //
DELIMITER ;
```

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

ticket_purchase_date	customer_id	class_id	complimentary_services(class_id)
26-12-2018	27	Economy	No
02-02-2020	22	Economy Plus	Yes
03-03-2020	21	Bussiness	No
04-04-2020	4	First Class	No
05-05-2020	5	Economy	No
07-07-2020	7	Bussiness	No
08-08-2020	8	Economy Plus	Yes
09-09-2020	9	First Class	No
10-10-2020	10	Economy	No
11-11-2020	11	Bussiness	No
12-12-2020	19	Economy Plus	Yes
01-01-2019	13	First Class	No
02-02-2019	14	Economy	No

Result 2

Output

Action Output

#	Time	Action	Message
68	22:43:51	CALL ticket_details_with_services()	50 row(s) returned

---

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.

```
DELIMITER //
CREATE PROCEDURE last_name_scott()
BEGIN
  DECLARE cust_id INT;
  DECLARE fst_name VARCHAR(128);
  DECLARE lst_name VARCHAR(128);
  DECLARE done INT DEFAULT 0;

  DECLARE find_scott CURSOR FOR
   SELECT customer_id, first_name, last_name
   FROM customer
   WHERE last_name LIKE '%Scott';

  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

  OPEN find_scott;
```

```
  read_loop: LOOP
    FETCH find_scott INTO cust_id, fst_name, lst_name;
    IF done = 1 THEN
      LEAVE read_loop;
    END IF;
    SELECT cust_id AS CustomerID,
        fst_name AS FirstName,
        lst_name AS LastName;
  END LOOP;
  CLOSE find_scott;
END //
DELIMITER ;
```

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

	CustomerID	FirstName	LastName
▶	38	Alexis	Scott

Result 7 | Result 8

Output

Action Output

	#	Time	Action	Message
✔	18	17:09:52	CALL last_name_scott()	1 row(s) returned