**SQL CODING CHALLENGE**

Name: S. Nikhil Sai

Date: 8-11-2024

CREATE TABLE burger\_names(

burger\_id INTEGER NOT NULL PRIMARY KEY

,burger\_name VARCHAR(10) NOT NULL

);

INSERT INTO burger\_names(burger\_id,burger\_name) VALUES (1,'Meatlovers');

INSERT INTO burger\_names(burger\_id,burger\_name) VALUES (2,'Vegetarian');

CREATE TABLE runner\_orders(

order\_id INTEGER NOT NULL PRIMARY KEY,

runner\_id INTEGER NOT NULL,

pickup\_time datetime,

distance VARCHAR(7),

duration VARCHAR(10),

cancellation VARCHAR(23),

);

INSERT INTO runner\_orders VALUES (1,1,'2021-01-01 18:15:34','20km','32 minutes',NULL);

INSERT INTO runner\_orders VALUES (2,1,'2021-01-01 19:10:54','20km','27 minutes',NULL);

INSERT INTO runner\_orders VALUES (3,1,'2021-01-03 00:12:37','13.4km','20 mins',NULL);

INSERT INTO runner\_orders VALUES (4,2,'2021-01-04 13:53:03','23.4','40',NULL);

INSERT INTO runner\_orders VALUES (5,3,'2021-01-08 21:10:57','10','15',NULL);

INSERT INTO runner\_orders VALUES (6,3,NULL,NULL,NULL,'Restaurant Cancellation');

INSERT INTO runner\_orders VALUES (7,2,'2021-01-08 21:30:45','25km','25mins',NULL);

INSERT INTO runner\_orders VALUES (8,2,'2021-01-10 00:15:02','23.4 km','15 minute',NULL);

INSERT INTO runner\_orders VALUES (9,2,NULL,NULL,NULL,'Customer Cancellation');

INSERT INTO runner\_orders VALUES (10,1,'2021-01-11 18:50:20','10km','10minutes',NULL);

select \* from runner\_orders

CREATE TABLE burger\_runner(

runner\_id INTEGER NOT NULL PRIMARY KEY

,registration\_date date NOT NULL

);

INSERT INTO burger\_runner VALUES (1,'2021-01-01');

INSERT INTO burger\_runner VALUES (2,'2021-01-03');

INSERT INTO burger\_runner VALUES (3,'2021-01-08');

INSERT INTO burger\_runner VALUES (4,'2021-01-15');

CREATE TABLE customer\_orders(

order\_id INTEGER NOT NULL

,customer\_id INTEGER NOT NULL

,burger\_id INTEGER NOT NULL

,exclusions VARCHAR(4)

,extras VARCHAR(4)

,order\_time datetime NOT NULL

);

INSERT INTO customer\_orders VALUES (1,101,1,NULL,NULL,'2021-01-01 18:05:02');

INSERT INTO customer\_orders VALUES (2,101,1,NULL,NULL,'2021-01-01 19:00:52');

INSERT INTO customer\_orders VALUES (3,102,1,NULL,NULL,'2021-01-02 23:51:23');

INSERT INTO customer\_orders VALUES (3,102,2,NULL,NULL,'2021-01-02 23:51:23');

INSERT INTO customer\_orders VALUES (4,103,1,'4',NULL,'2021-01-04 13:23:46');

INSERT INTO customer\_orders VALUES (4,103,1,'4',NULL,'2021-01-04 13:23:46');

INSERT INTO customer\_orders VALUES (4,103,2,'4',NULL,'2021-01-04 13:23:46');

INSERT INTO customer\_orders VALUES (5,104,1,NULL,'1','2021-01-08 21:00:29');

INSERT INTO customer\_orders VALUES (6,101,2,NULL,NULL,'2021-01-08 21:03:13');

INSERT INTO customer\_orders VALUES (7,105,2,NULL,'1','2021-01-08 21:20:29');

INSERT INTO customer\_orders VALUES (8,102,1,NULL,NULL,'2021-01-09 23:54:33');

INSERT INTO customer\_orders VALUES (9,103,1,'4','1, 5','2021-01-10 11:22:59');

INSERT INTO customer\_orders VALUES (10,104,1,NULL,NULL,'2021-01-11 18:34:49');

INSERT INTO customer\_orders VALUES (10,104,1,'2, 6','1, 4','2021-01-11 18:34:49');

1. **Querying Data by Using Joins and Subqueries & subtotal:**

**1.1** SELECT customer\_id, COUNT(order\_id) AS total\_orders

FROM customer\_orders

GROUP BY customer\_id;

A screenshot of a computer

Description automatically generated

**Explanation:** This query counts the total orders placed by each customer by grouping on customer\_id by using GROUP BY and using COUNT to count the total order\_id values.

**1.2** SELECT DISTINCT r.runner\_id, r.registration\_date

FROM burger\_runner r

JOIN runner\_orders ro ON r.runner\_id = ro.runner\_id

WHERE ro.cancellation IS NULL;

A screenshot of a computer program

Description automatically generated

**Explanation:** This query inner joins burger\_runner and runner\_orders tables to identify runners who have completed a delivery where cancellation is NULL and lists them without repeatition.

**1.3** SELECT co.customer\_id, co.order\_id, co.order\_time

FROM customer\_orders co

WHERE co.order\_time = (

SELECT MAX(order\_time) FROM customer\_orders WHERE customer\_id = co.customer\_id);

A screenshot of a computer

Description automatically generated

**Explanation:** This subquery finds each customer's latest order time using MAX aggregate function. The main query then retrieves each customer's latest order details by matching order\_time.

**1.4** SELECT DISTINCT co.customer\_id

FROM customer\_orders co

JOIN burger\_names bn ON co.burger\_id = bn.burger\_id

WHERE bn.burger\_name = 'Meatlovers';

A screenshot of a computer

Description automatically generated

**Explanation:** This query inner joins customer\_orders and burger\_names to match orders of the "Meatlovers" burger. It uses DISTINCT to list each unique customer who ordered this burger.

**1.5** SELECT customer\_id, MAX(order\_time) AS latest\_order

FROM customer\_orders

GROUP BY customer\_id;

A screenshot of a computer

Description automatically generated

* Explanation: This query retrives the latest order placed by each customer by grouping on customer\_id and using MAX aggregate function to get the most recent order\_time.

1. **Manipulate data by using sql commands using groupby and having clause.**

**2.1** SELECT customer\_id, COUNT(order\_id) AS total\_orders

FROM customer\_orders

GROUP BY customer\_id

HAVING COUNT(order\_id) >= 3;

A screenshot of a computer

Description automatically generated

**Explanation**: This query counts each customer’s total orders grouped by customer\_id and then filters with HAVING to show only customers who have placed more than or equal to three orders.

**2.2** SELECT runner\_id, COUNT(order\_id) AS successful\_orders

FROM runner\_orders

WHERE cancellation IS NULL

GROUP BY runner\_id

HAVING COUNT(order\_id) > 2;

A screenshot of a computer

Description automatically generated

**Explanation:** This query counts each runner’s completed orders (non-canceled) and uses HAVING to show only runners with more than two successful orders.