CREATE TEMP FUNCTION parseHours(input STRING)

RETURNS ARRAY<STRUCT<day STRING, start\_time STRING, end\_time STRING>>

LANGUAGE js AS """

try {

const parsed = JSON.parse(input);

const menus = parsed.data.menus || {};

const menuKey = Object.keys(menus)[0];

const menu = menus[menuKey];

const sections = menu.sections || [];

const section = sections[0] || {};

const regularHours = section.regularHours || [];

const days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'];

const result = [];

regularHours.forEach(hours => {

const daysBitArray = hours.daysBitArray || [];

const startTime = hours.startTime || null;

const endTime = hours.endTime || null;

for (let i = 0; i < days.length; i++) {

if (daysBitArray[i] === true) {

result.push({

day: days[i],

start\_time: startTime,

end\_time: endTime

});

}

}

});

return result;

} catch (e) {

return [];

}

""";

WITH parsed\_hours AS (

SELECT

ue\_slug,

vb\_name,

b\_name,

timestamp,

day\_time.day AS day,

time(datetime(concat(CURRENT\_DATE(), " ", day\_time.start\_time, ':00'))) AS start\_time,

time(datetime(concat(CURRENT\_DATE(), " ", day\_time.end\_time, ':00'))) AS end\_time

FROM (

SELECT

slug AS ue\_slug,

vb\_name,

b\_name,

timestamp,

ARRAY(SELECT AS STRUCT day, start\_time, end\_time FROM UNNEST(parseHours(TO\_JSON\_STRING(response)))) AS day\_times

FROM (

SELECT \* ,

ROW\_NUMBER() OVER (PARTITION BY slug ORDER BY timestamp DESC) AS rn

FROM arboreal-vision-339901.take\_home\_v2.virtual\_kitchen\_ubereats\_hours

)

WHERE rn = 1

) CROSS JOIN UNNEST(day\_times) AS day\_time

), deduplicated\_base AS (

SELECT DISTINCT

slug,

timestamp,

vb\_name,

b\_name,

JSON\_EXTRACT\_SCALAR(value, '$.days\_of\_week[0]') AS day,

JSON\_EXTRACT\_SCALAR(value, '$.from') AS open\_time,

JSON\_EXTRACT\_SCALAR(value, '$.to') AS close\_time

FROM arboreal-vision-339901.take\_home\_v2.virtual\_kitchen\_grubhub\_hours,

UNNEST(JSON\_QUERY\_ARRAY(response, '$.availability\_by\_catalog.STANDARD\_DELIVERY.schedule\_rules')) AS value

), latest\_timestamp AS (

SELECT slug, MAX(timestamp) AS latest\_ts

FROM deduplicated\_base

GROUP BY slug

), daily\_hours\_calc AS (

SELECT

d.\*

FROM deduplicated\_base d

INNER JOIN latest\_timestamp lt

ON d.slug = lt.slug

AND d.timestamp = lt.latest\_ts

), combined\_hours AS (

SELECT

gru.slug AS Grubhub\_slug,

ue.ue\_slug AS Uber\_Eats\_slug,

gru.vb\_name,

gru.b\_name,

-- Combine Grubhub business hours for all days

STRING\_AGG(CONCAT(gru.day, ': ', gru.open\_time, '-', gru.close\_time), '; ') AS Grubhub\_Business\_Hours,

-- Combine Uber Eats business hours for all days

STRING\_AGG(CONCAT(ue.day, ': ', ue.start\_time, '-', ue.end\_time), '; ') AS Uber\_Eats\_Business\_Hours,

-- Determine the overall status for is\_out\_range

CASE

WHEN COUNT(DISTINCT

CASE

WHEN LEAST(

ABS(TIMESTAMP\_DIFF(TIMESTAMP(DATETIME(CURRENT\_DATE(), ue.start\_time)), TIMESTAMP(DATETIME(CURRENT\_DATE(), CAST(gru.open\_time AS TIME))), SECOND) / 60),

1440 - ABS(TIMESTAMP\_DIFF(TIMESTAMP(DATETIME(CURRENT\_DATE(), ue.start\_time)), TIMESTAMP(DATETIME(CURRENT\_DATE(), CAST(gru.open\_time AS TIME))), SECOND) / 60)

) = 0

AND LEAST(

ABS(TIMESTAMP\_DIFF(TIMESTAMP(DATETIME(CURRENT\_DATE(), CAST(gru.close\_time AS TIME))), TIMESTAMP(DATETIME(CURRENT\_DATE(), ue.end\_time)), SECOND) / 60),

1440 - ABS(TIMESTAMP\_DIFF(TIMESTAMP(DATETIME(CURRENT\_DATE(), CAST(gru.close\_time AS TIME))), TIMESTAMP(DATETIME(CURRENT\_DATE(), ue.end\_time)), SECOND) / 60)

) = 0 THEN 'In Range'

WHEN LEAST(

ABS(TIMESTAMP\_DIFF(TIMESTAMP(DATETIME(CURRENT\_DATE(), ue.start\_time)), TIMESTAMP(DATETIME(CURRENT\_DATE(), CAST(gru.open\_time AS TIME))), SECOND) / 60),

1440 - ABS(TIMESTAMP\_DIFF(TIMESTAMP(DATETIME(CURRENT\_DATE(), ue.start\_time)), TIMESTAMP(DATETIME(CURRENT\_DATE(), CAST(gru.open\_time AS TIME))), SECOND) / 60)

) <= 5

AND LEAST(

ABS(TIMESTAMP\_DIFF(TIMESTAMP(DATETIME(CURRENT\_DATE(), CAST(gru.close\_time AS TIME))), TIMESTAMP(DATETIME(CURRENT\_DATE(), ue.end\_time)), SECOND) / 60),

1440 - ABS(TIMESTAMP\_DIFF(TIMESTAMP(DATETIME(CURRENT\_DATE(), CAST(gru.close\_time AS TIME))), TIMESTAMP(DATETIME(CURRENT\_DATE(), ue.end\_time)), SECOND) / 60)

) <= 5 THEN 'Out of Range with 5 mins difference'

ELSE 'Out of Range'

END

) = 1 THEN ARRAY\_AGG(DISTINCT

CASE

WHEN LEAST(

ABS(TIMESTAMP\_DIFF(TIMESTAMP(DATETIME(CURRENT\_DATE(), ue.start\_time)), TIMESTAMP(DATETIME(CURRENT\_DATE(), CAST(gru.open\_time AS TIME))), SECOND) / 60),

1440 - ABS(TIMESTAMP\_DIFF(TIMESTAMP(DATETIME(CURRENT\_DATE(), ue.start\_time)), TIMESTAMP(DATETIME(CURRENT\_DATE(), CAST(gru.open\_time AS TIME))), SECOND) / 60)

) = 0

AND LEAST(

ABS(TIMESTAMP\_DIFF(TIMESTAMP(DATETIME(CURRENT\_DATE(), CAST(gru.close\_time AS TIME))), TIMESTAMP(DATETIME(CURRENT\_DATE(), ue.end\_time)), SECOND) / 60),

1440 - ABS(TIMESTAMP\_DIFF(TIMESTAMP(DATETIME(CURRENT\_DATE(), CAST(gru.close\_time AS TIME))), TIMESTAMP(DATETIME(CURRENT\_DATE(), ue.end\_time)), SECOND) / 60)

) = 0 THEN 'In Range'

WHEN LEAST(

ABS(TIMESTAMP\_DIFF(TIMESTAMP(DATETIME(CURRENT\_DATE(), ue.start\_time)), TIMESTAMP(DATETIME(CURRENT\_DATE(), CAST(gru.open\_time AS TIME))), SECOND) / 60),

1440 - ABS(TIMESTAMP\_DIFF(TIMESTAMP(DATETIME(CURRENT\_DATE(), ue.start\_time)), TIMESTAMP(DATETIME(CURRENT\_DATE(), CAST(gru.open\_time AS TIME))), SECOND) / 60)

) <= 5

AND LEAST(

ABS(TIMESTAMP\_DIFF(TIMESTAMP(DATETIME(CURRENT\_DATE(), CAST(gru.close\_time AS TIME))), TIMESTAMP(DATETIME(CURRENT\_DATE(), ue.end\_time)), SECOND) / 60),

1440 - ABS(TIMESTAMP\_DIFF(TIMESTAMP(DATETIME(CURRENT\_DATE(), CAST(gru.close\_time AS TIME))), TIMESTAMP(DATETIME(CURRENT\_DATE(), ue.end\_time)), SECOND) / 60)

) <= 5 THEN 'Out of Range with 5 mins difference'

ELSE 'Out of Range'

END LIMIT 1

)[OFFSET(0)]

ELSE 'Mixed Statuses'

END AS is\_out\_range

FROM daily\_hours\_calc gru

INNER JOIN parsed\_hours ue

ON gru.vb\_name = ue.vb\_name

AND gru.b\_name = ue.b\_name

AND LOWER(gru.day) = LOWER(ue.day)

GROUP BY gru.slug, ue.ue\_slug, gru.vb\_name, gru.b\_name

)

SELECT

Grubhub\_slug,

Grubhub\_Business\_Hours AS Virtual\_Restaurant\_Business\_Hours,

Uber\_Eats\_slug,

Uber\_Eats\_Business\_Hours,

is\_out\_range

FROM combined\_hours;