Q1. Write an SQL query to find the names of restaurants that have at least one menu item with a price greater than $10.

Solution:

```
SELECT
    R.name
FROM
    restaurant_info R
        INNER JOIN
    (SELECT
        restaurant_id
    FROM
        menuitems
    WHERE
        PRICE > 10) AS REST ON R.restaurant_id =
REST.restaurant_id
GROUP BY R.restaurant_id
ORDER BY R.name;
```

Q2. Write an SQL query to retrieve the user names and their corresponding orders where the order total is greater than the average order total for all users.

Solution:

```
SELECT
    U.name AS USER_NAME, ORDERS.order_id AS ORDER_ID
FROM
    user_info U
        INNER JOIN
```

```
    (SELECT
        user_id, order_id
    FROM
        orders
    WHERE
        total_amount > (SELECT AVG(total_amount) FROM
orders)) AS ORDERS ON U.ID = ORDERS.user_id
ORDER BY 1;
```

Q3. Write an SQL query to list the names of users whose last names start with 'S' or ends with ‘e’.

Solution:

```
SELECT
    NAME
FROM
    user_info
WHERE
    SUBSTRING(name, LENGTH(SUBSTRING_INDEX(name, ' ', 1)) +
2) LIKE 'S%'
        OR SUBSTRING(name, LENGTH(SUBSTRING_INDEX(name, ' ',
1)) + 2) LIKE '%e';
```

Q4. Write an SQL query to find the total order amounts for each restaurant. If a restaurant has no orders, display the restaurant name and a total amount of 0. Use the COALESCE function to handle null values.

Solution:

```
SELECT
    R.name AS RESTAURANT,
    ROUND(COALESCE(SUM(total_amount), 0), 2)
TOTAL_ORDER_AMOUNTS
FROM
    restaurant_info R
        LEFT JOIN
    orders O ON R.restaurant_id = O.restaurant_id
GROUP BY R.restaurant_id;
```

Q5. Write a query to find out how many orders were placed using cash or credit.

Solution:

```
SELECT
    T.name AS PAYMENT_TYPE, COUNT(P.order_id) AS
ORDERS_COUNT
FROM
    payment_transactions P
        INNER JOIN
    payment_type T ON P.pay_type_id = T.pay_type_id
GROUP BY P.pay_type_id;
```