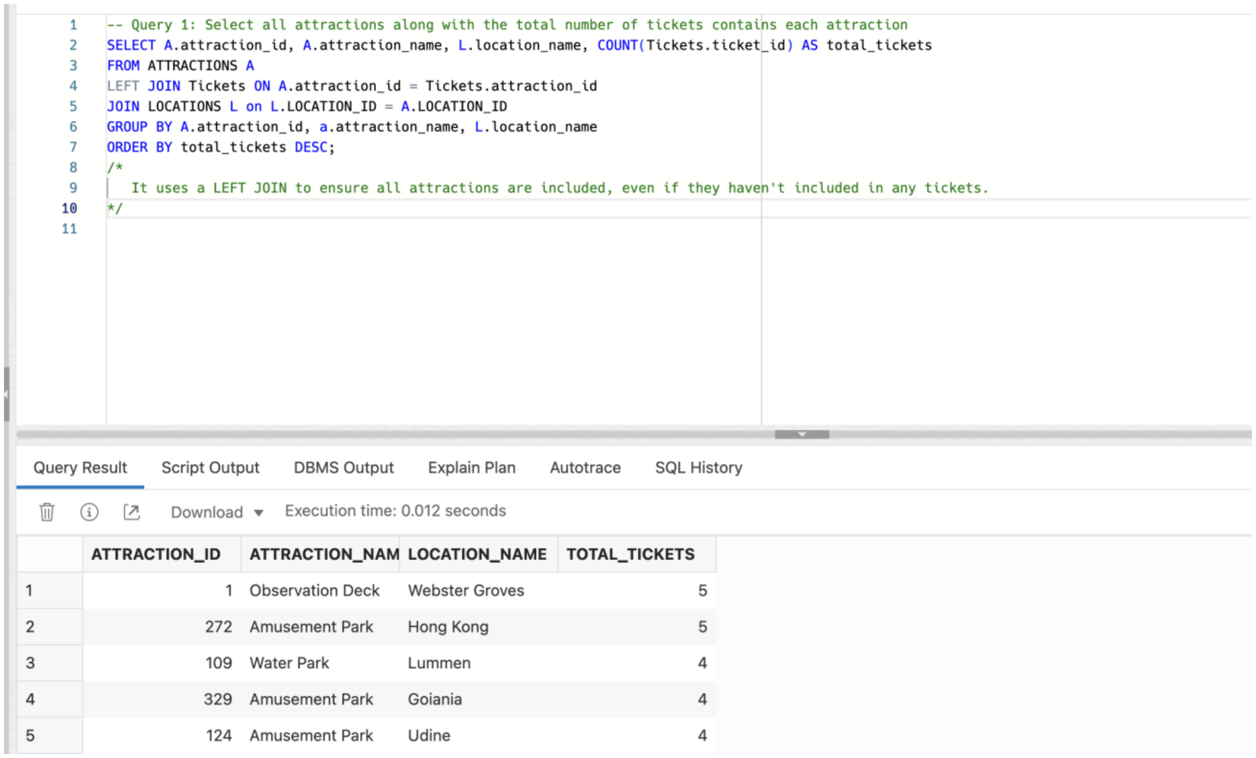
## Step 2 – Queries

Pay attention, I changed the queries to use List with select in the SQL files!

Query 1:



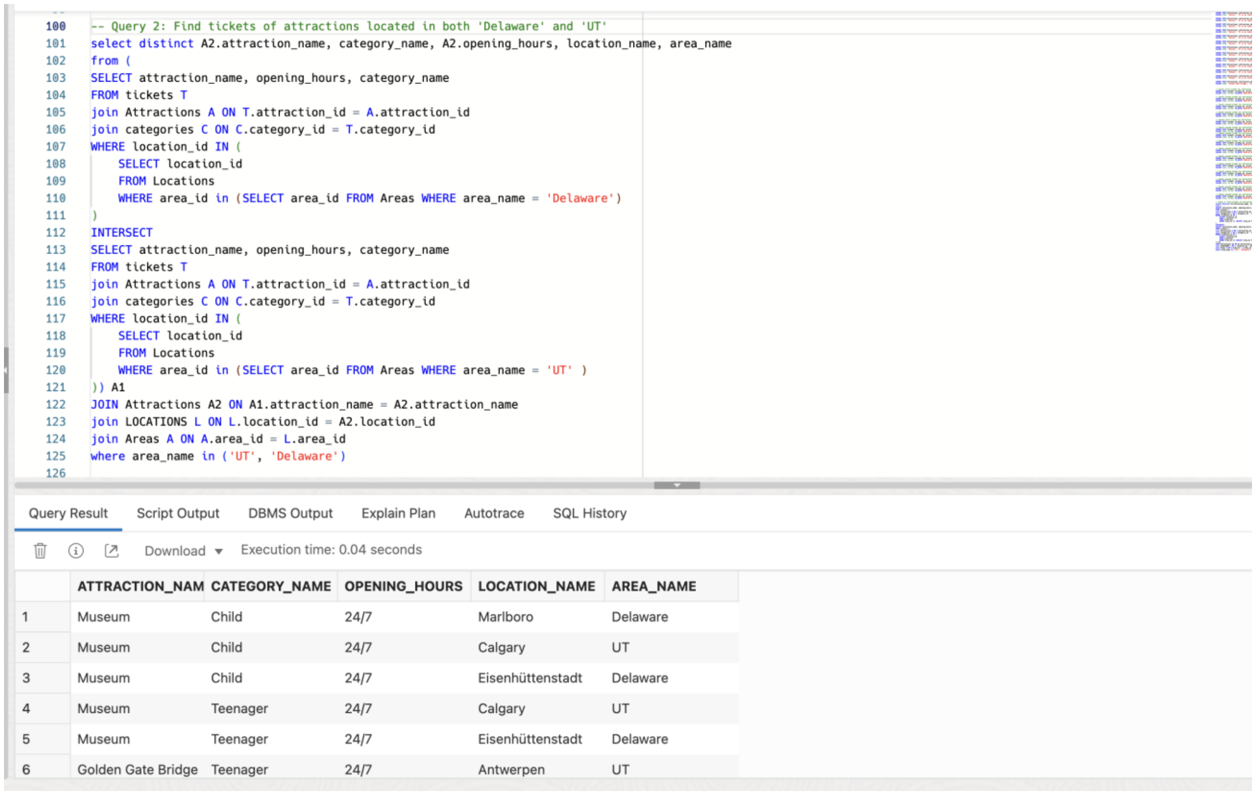
```
-- Query 1: Select all attractions along with the total number of tickets contains each attraction
SELECT A.attraction_id, A.attraction_name, L.location_name, COUNT(Tickets.ticket_id) AS total_tickets
FROM ATTRACTIONS A
LEFT JOIN Tickets ON A.attraction_id = Tickets.attraction_id
JOIN LOCATIONS L on L.LOCATION_ID = A.LOCATION_ID
GROUP BY A.attraction_id, a.attraction_name, L.location_name
ORDER BY total_tickets DESC;
/*
   It uses a LEFT JOIN to ensure all attractions are included, even if they haven't included in any tickets.
*/
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download ▾ Execution time: 0.012 seconds

| | ATTRACTION_ID | ATTRACTION_NAM | LOCATION_NAME | TOTAL_TICKETS |
|---|---|---|---|---|
| 1 | 1 | Observation Deck | Webster Groves | 5 |
| 2 | 272 | Amusement Park | Hong Kong | 5 |
| 3 | 109 | Water Park | Lummen | 4 |
| 4 | 329 | Amusement Park | Goiania | 4 |
| 5 | 124 | Amusement Park | Udine | 4 |

Query 2:

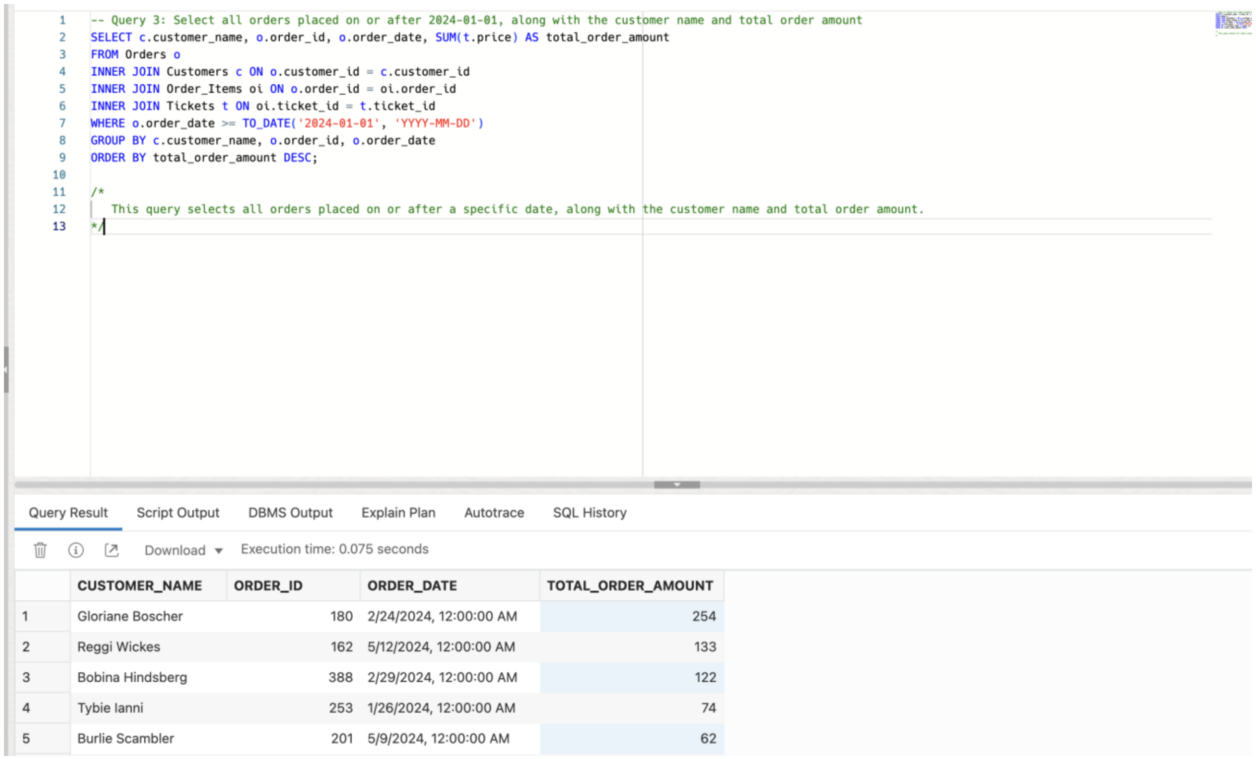


```sql
-- Query 2: Find tickets of attractions located in both 'Delaware' and 'UT'
select distinct A2.attraction_name, category_name, A2.opening_hours, location_name, area_name
from (
SELECT attraction_name, opening_hours, category_name
FROM tickets T
join Attractions A ON T.attraction_id = A.attraction_id
join categories C ON C.category_id = T.category_id
WHERE location_id IN (
    SELECT location_id
    FROM Locations
    WHERE area_id in (SELECT area_id FROM Areas WHERE area_name = 'Delaware')
)
INTERSECT
SELECT attraction_name, opening_hours, category_name
FROM tickets T
join Attractions A ON T.attraction_id = A.attraction_id
join categories C ON C.category_id = T.category_id
WHERE location_id IN (
    SELECT location_id
    FROM Locations
    WHERE area_id in (SELECT area_id FROM Areas WHERE area_name = 'UT' )
)) A1
JOIN Attractions A2 ON A1.attraction_name = A2.attraction_name
join LOCATIONS L ON L.location_id = A2.location_id
join Areas A ON A.area_id = L.area_id
where area_name in ('UT', 'Delaware')
```

Execution time: 0.04 seconds

| | ATTRACTION_NAM | CATEGORY_NAME | OPENING_HOURS | LOCATION_NAME | AREA_NAME |
|---|---|---|---|---|---|
| 1 | Museum | Child | 24/7 | Marlboro | Delaware |
| 2 | Museum | Child | 24/7 | Calgary | UT |
| 3 | Museum | Child | 24/7 | Eisenhüttenstadt | Delaware |
| 4 | Museum | Teenager | 24/7 | Calgary | UT |
| 5 | Museum | Teenager | 24/7 | Eisenhüttenstadt | Delaware |
| 6 | Golden Gate Bridge | Teenager | 24/7 | Antwerpen | UT |

Query 3:

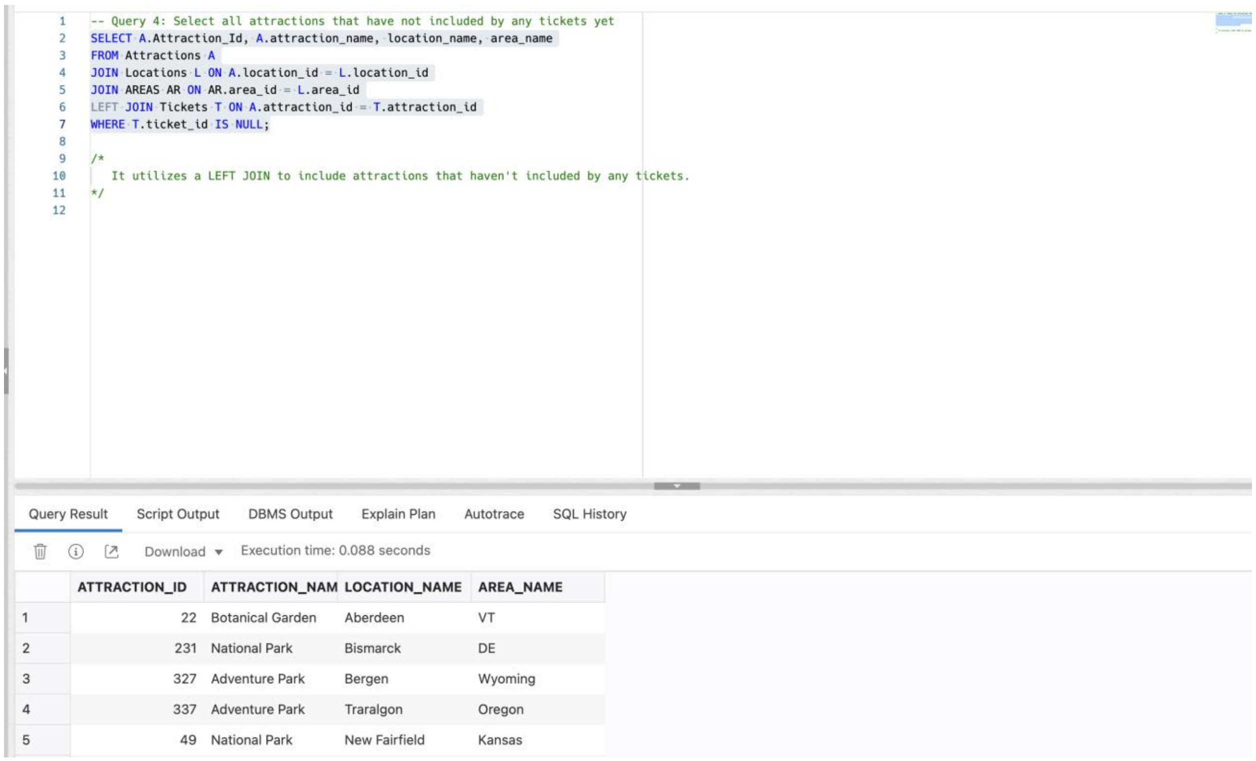


```sql
-- Query 3: Select all orders placed on or after 2024-01-01, along with the customer name and total order amount
SELECT c.customer_name, o.order_id, o.order_date, SUM(t.price) AS total_order_amount
FROM Orders o
INNER JOIN Customers c ON o.customer_id = c.customer_id
INNER JOIN Order_Items oi ON o.order_id = oi.order_id
INNER JOIN Tickets t ON oi.ticket_id = t.ticket_id
WHERE o.order_date >= TO_DATE('2024-01-01', 'YYYY-MM-DD')
GROUP BY c.customer_name, o.order_id, o.order_date
ORDER BY total_order_amount DESC;

/*
   This query selects all orders placed on or after a specific date, along with the customer name and total order amount.
*/
```

Execution time: 0.075 seconds

| | CUSTOMER_NAME | ORDER_ID | ORDER_DATE | TOTAL_ORDER_AMOUNT |
|---|---|---|---|---|
| 1 | Gloriane Boscher | 180 | 2/24/2024, 12:00:00 AM | 254 |
| 2 | Reggi Wickes | 162 | 5/12/2024, 12:00:00 AM | 133 |
| 3 | Bobina Hindsberg | 388 | 2/29/2024, 12:00:00 AM | 122 |
| 4 | Tybie Ianni | 253 | 1/26/2024, 12:00:00 AM | 74 |
| 5 | Burlie Scambler | 201 | 5/9/2024, 12:00:00 AM | 62 |

Query 4:



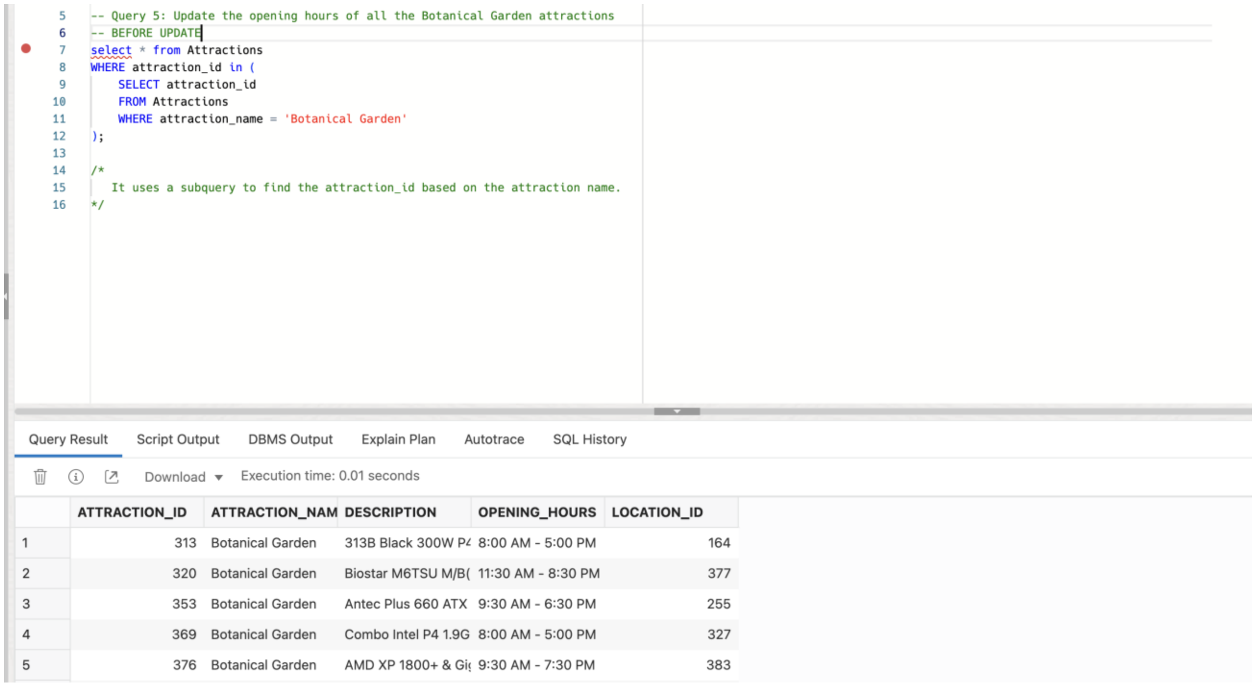
```sql
-- Query 4: Select all attractions that have not included by any tickets yet
SELECT A.Attraction_Id, A.attraction_name, location_name, area_name
FROM Attractions A
JOIN Locations L ON A.location_id = L.location_id
JOIN AREAS AR ON AR.area_id = L.area_id
LEFT JOIN Tickets T ON A.attraction_id = T.attraction_id
WHERE T.ticket_id IS NULL;

/*
    It utilizes a LEFT JOIN to include attractions that haven't included by any tickets.
*/
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

Download ▾ Execution time: 0.088 seconds

| | ATTRACTION_ID | ATTRACTION_NAM | LOCATION_NAME | AREA_NAME |
|---|---|---|---|---|
| 1 | 22 | Botanical Garden | Aberdeen | VT |
| 2 | 231 | National Park | Bismarck | DE |
| 3 | 327 | Adventure Park | Bergen | Wyoming |
| 4 | 337 | Adventure Park | Traralgon | Oregon |
| 5 | 49 | National Park | New Fairfield | Kansas |

Query 5 – Before update:



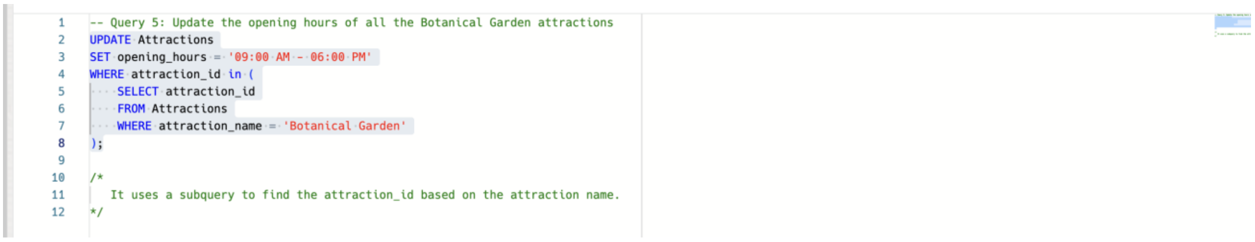
```sql
-- Query 5: Update the opening hours of all the Botanical Garden attractions
-- BEFORE UPDATE
select * from Attractions
WHERE attraction_id in (
    SELECT attraction_id
    FROM Attractions
    WHERE attraction_name = 'Botanical Garden'
);

/*
    It uses a subquery to find the attraction_id based on the attraction name.
*/
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History

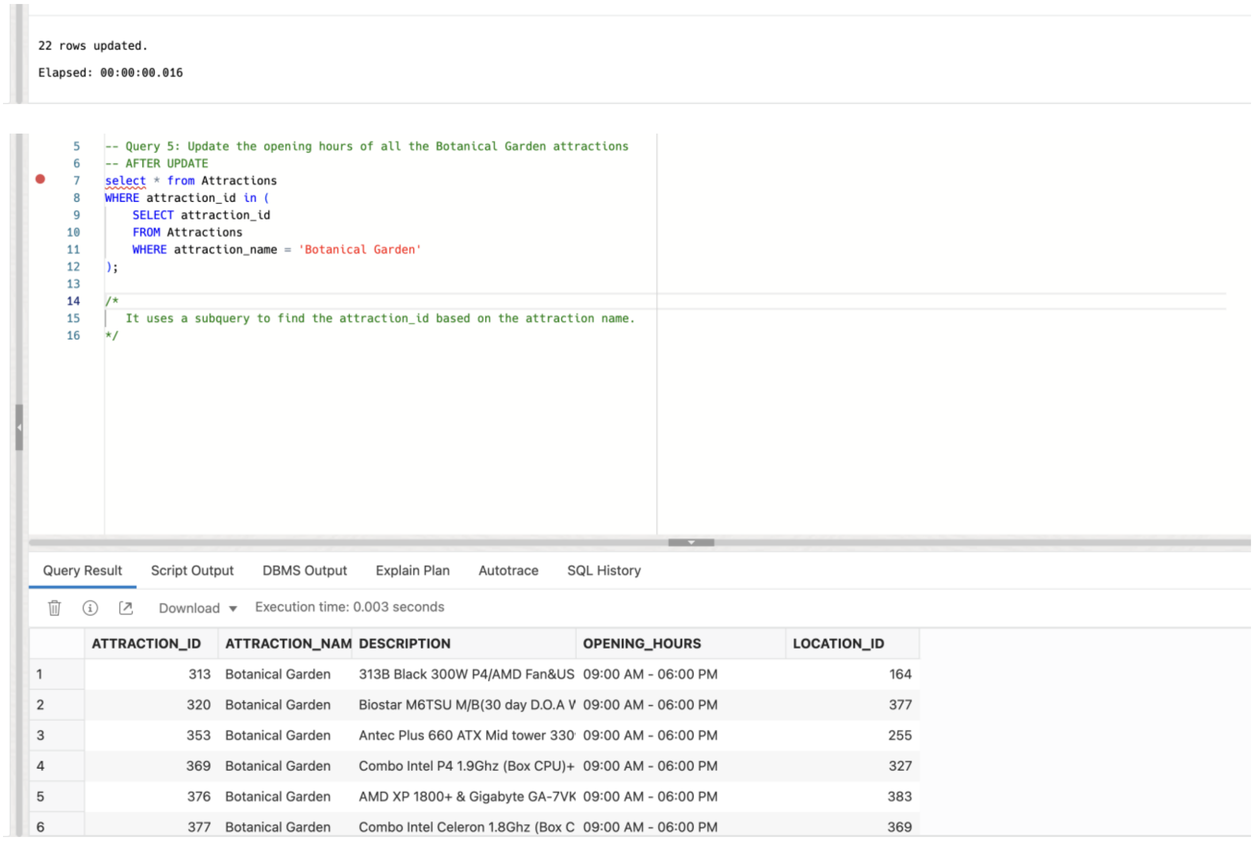
Download ▾ Execution time: 0.01 seconds

| | ATTRACTION_ID | ATTRACTION_NAM | DESCRIPTION | OPENING_HOURS | LOCATION_ID |
|---|---|---|---|---|---|
| 1 | 313 | Botanical Garden | 313B Black 300W P4 | 8:00 AM - 5:00 PM | 164 |
| 2 | 320 | Botanical Garden | Biostar M6TSU M/B( | 11:30 AM - 8:30 PM | 377 |
| 3 | 353 | Botanical Garden | Antec Plus 660 ATX | 9:30 AM - 6:30 PM | 255 |
| 4 | 369 | Botanical Garden | Combo Intel P4 1.9G | 8:00 AM - 5:00 PM | 327 |
| 5 | 376 | Botanical Garden | AMD XP 1800+ & Gi( | 9:30 AM - 7:30 PM | 383 |

Query 5 – Update:


```
-- Query 5: Update the opening hours of all the Botanical Garden attractions
UPDATE Attractions
SET opening_hours = '09:00 AM - 06:00 PM'
WHERE attraction_id in (
    SELECT attraction_id
    FROM Attractions
    WHERE attraction_name = 'Botanical Garden'
);

/*
  It uses a subquery to find the attraction_id based on the attraction name.
*/
```

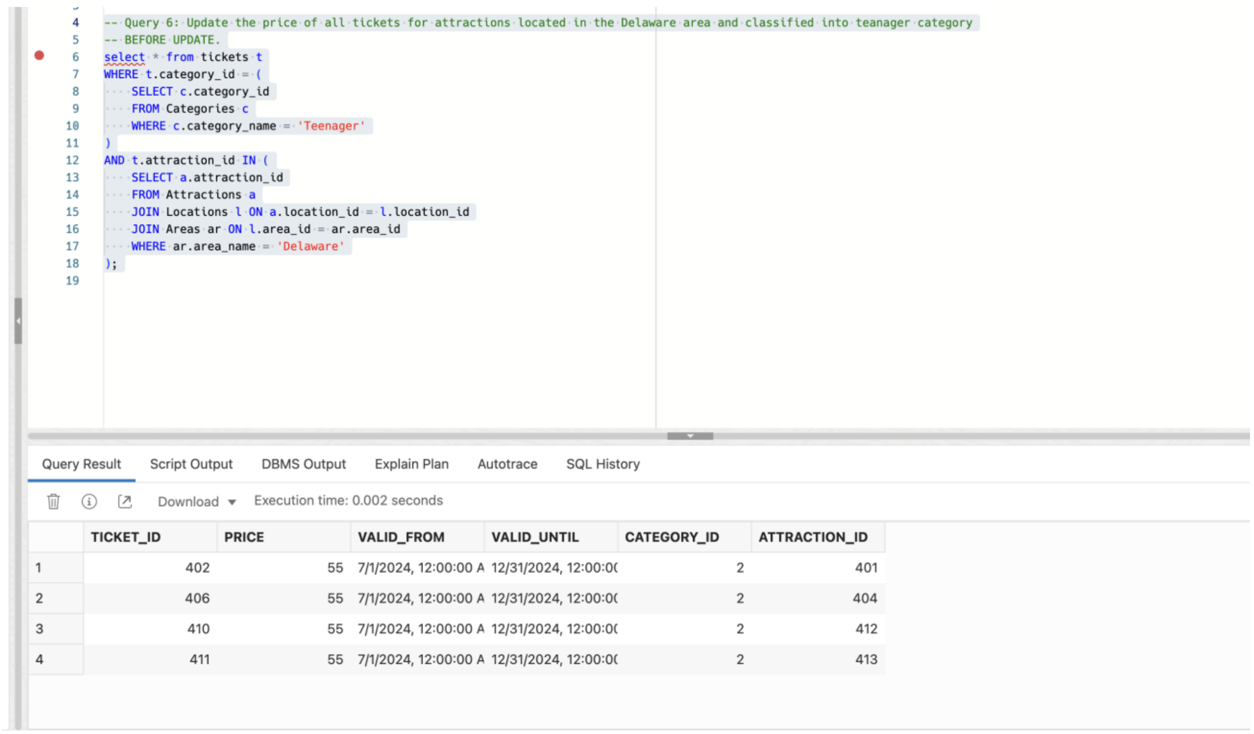
Query 5 – After update:


```
22 rows updated.

Elapsed: 00:00:00.016
```

```
-- Query 5: Update the opening hours of all the Botanical Garden attractions
-- AFTER UPDATE
select * from Attractions
WHERE attraction_id in (
    SELECT attraction_id
    FROM Attractions
    WHERE attraction_name = 'Botanical Garden'
);

/*
  It uses a subquery to find the attraction_id based on the attraction name.
*/
```

| | ATTRACTION_ID | ATTRACTION_NAM | DESCRIPTION | OPENING_HOURS | LOCATION_ID |
|---|---|---|---|---|---|
| 1 | 313 | Botanical Garden | 313B Black 300W P4/AMD Fan&US | 09:00 AM - 06:00 PM | 164 |
| 2 | 320 | Botanical Garden | Biostar M6TSU M/B(30 day D.O.A V | 09:00 AM - 06:00 PM | 377 |
| 3 | 353 | Botanical Garden | Antec Plus 660 ATX Mid tower 330 | 09:00 AM - 06:00 PM | 255 |
| 4 | 369 | Botanical Garden | Combo Intel P4 1.9Ghz (Box CPU)+ | 09:00 AM - 06:00 PM | 327 |
| 5 | 376 | Botanical Garden | AMD XP 1800+ & Gigabyte GA-7VK | 09:00 AM - 06:00 PM | 383 |
| 6 | 377 | Botanical Garden | Combo Intel Celeron 1.8Ghz (Box C | 09:00 AM - 06:00 PM | 369 |

Query 6 – Before update:

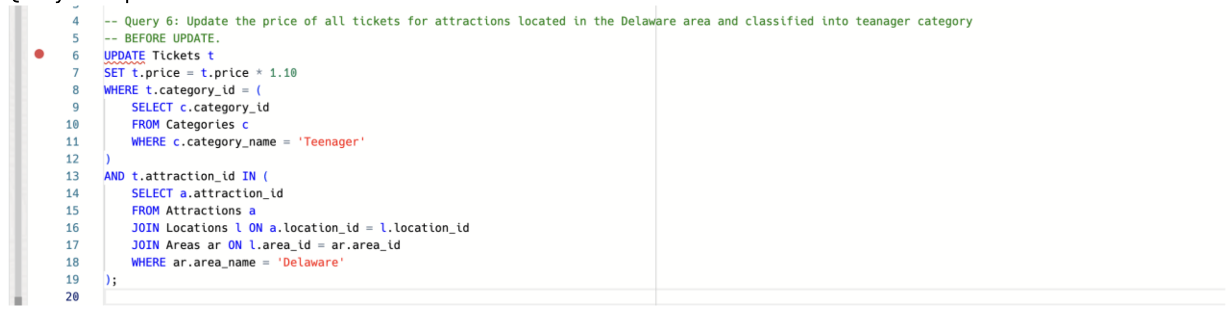


```sql
-- Query 6: Update the price of all tickets for attractions located in the Delaware area and classified into teanager category
-- BEFORE UPDATE.
select * from tickets t
WHERE t.category_id = (
    SELECT c.category_id
    FROM Categories c
    WHERE c.category_name = 'Teenager'
)
AND t.attraction_id IN (
    SELECT a.attraction_id
    FROM Attractions a
    JOIN Locations l ON a.location_id = l.location_id
    JOIN Areas ar ON l.area_id = ar.area_id
    WHERE ar.area_name = 'Delaware'
);
```

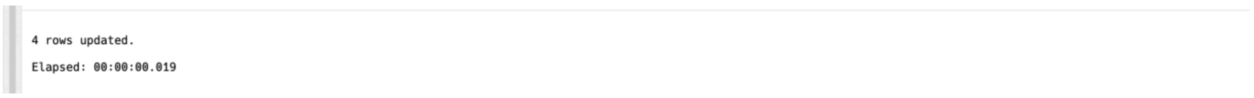
Execution time: 0.002 seconds

| | TICKET_ID | PRICE | VALID_FROM | VALID_UNTIL | CATEGORY_ID | ATTRACTION_ID |
|---|---|---|---|---|---|---|
| 1 | 402 | 55 | 7/1/2024, 12:00:00 A | 12/31/2024, 12:00:0( | 2 | 401 |
| 2 | 406 | 55 | 7/1/2024, 12:00:00 A | 12/31/2024, 12:00:0( | 2 | 404 |
| 3 | 410 | 55 | 7/1/2024, 12:00:00 A | 12/31/2024, 12:00:0( | 2 | 412 |
| 4 | 411 | 55 | 7/1/2024, 12:00:00 A | 12/31/2024, 12:00:0( | 2 | 413 |

Query 6 – Update:

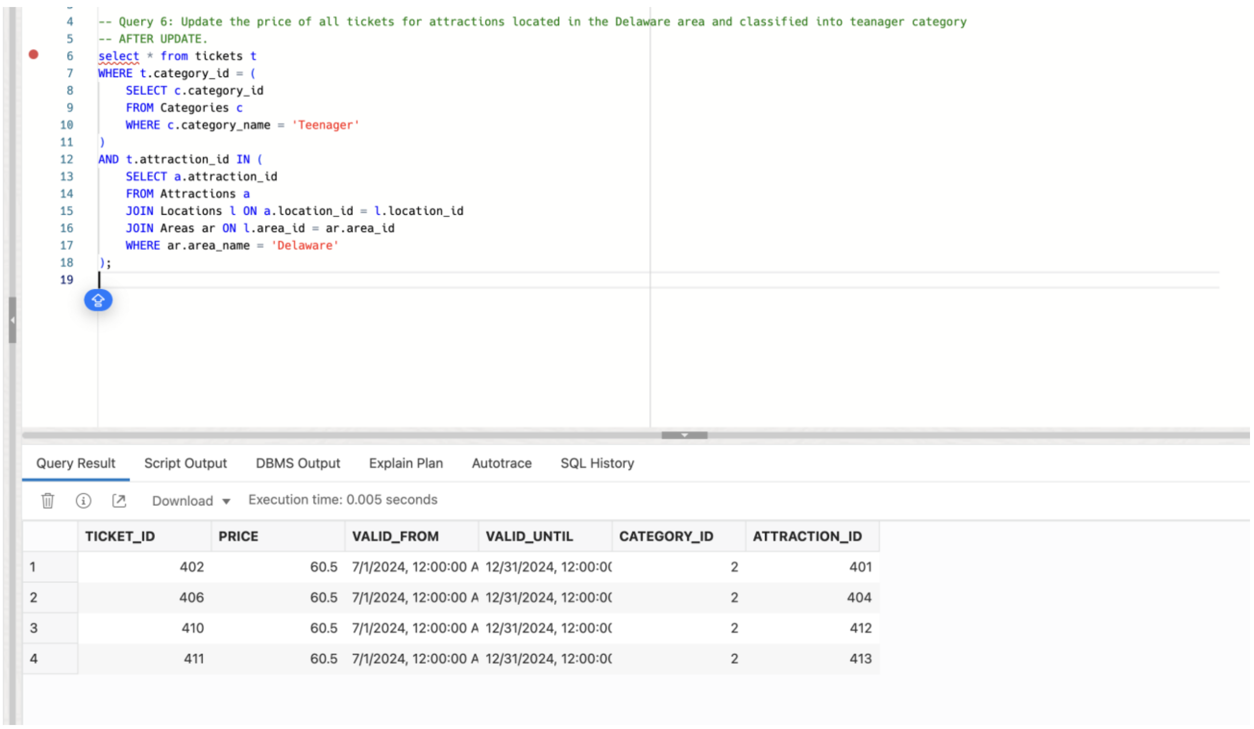


```sql
-- Query 6: Update the price of all tickets for attractions located in the Delaware area and classified into teanager category
-- BEFORE UPDATE.
UPDATE Tickets t
SET t.price = t.price * 1.10
WHERE t.category_id = (
    SELECT c.category_id
    FROM Categories c
    WHERE c.category_name = 'Teenager'
)
AND t.attraction_id IN (
    SELECT a.attraction_id
    FROM Attractions a
    JOIN Locations l ON a.location_id = l.location_id
    JOIN Areas ar ON l.area_id = ar.area_id
    WHERE ar.area_name = 'Delaware'
);
```



```
4 rows updated.

Elapsed: 00:00:00.019
```

Query 6 – After update:

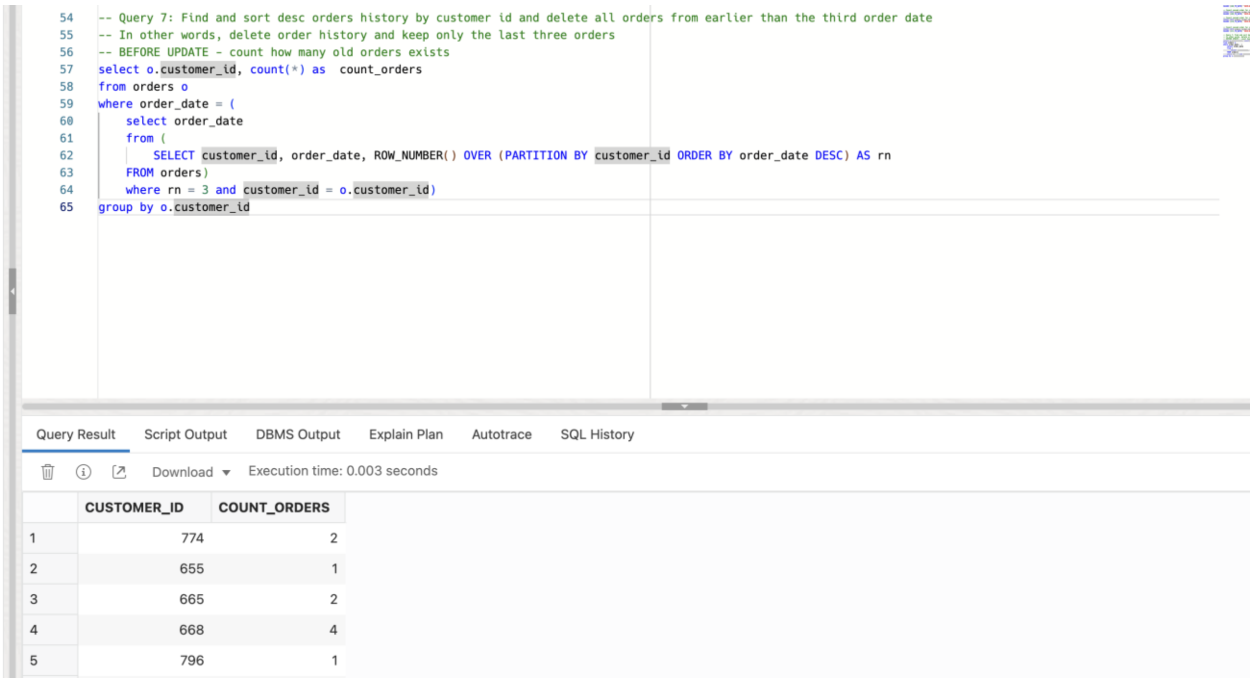


```sql
-- Query 6: Update the price of all tickets for attractions located in the Delaware area and classified into teanager category
-- AFTER UPDATE.
select * from tickets t
WHERE t.category_id = (
    SELECT c.category_id
    FROM Categories c
    WHERE c.category_name = 'Teenager'
)
AND t.attraction_id IN (
    SELECT a.attraction_id
    FROM Attractions a
    JOIN Locations l ON a.location_id = l.location_id
    JOIN Areas ar ON l.area_id = ar.area_id
    WHERE ar.area_name = 'Delaware'
);
```

Execution time: 0.005 seconds

| | TICKET_ID | PRICE | VALID_FROM | VALID_UNTIL | CATEGORY_ID | ATTRACTION_ID |
|---|---|---|---|---|---|---|
| 1 | 402 | 60.5 | 7/1/2024, 12:00:00 A | 12/31/2024, 12:00:0 | 2 | 401 |
| 2 | 406 | 60.5 | 7/1/2024, 12:00:00 A | 12/31/2024, 12:00:0 | 2 | 404 |
| 3 | 410 | 60.5 | 7/1/2024, 12:00:00 A | 12/31/2024, 12:00:0 | 2 | 412 |
| 4 | 411 | 60.5 | 7/1/2024, 12:00:00 A | 12/31/2024, 12:00:0 | 2 | 413 |

Query 7 – Before delete:

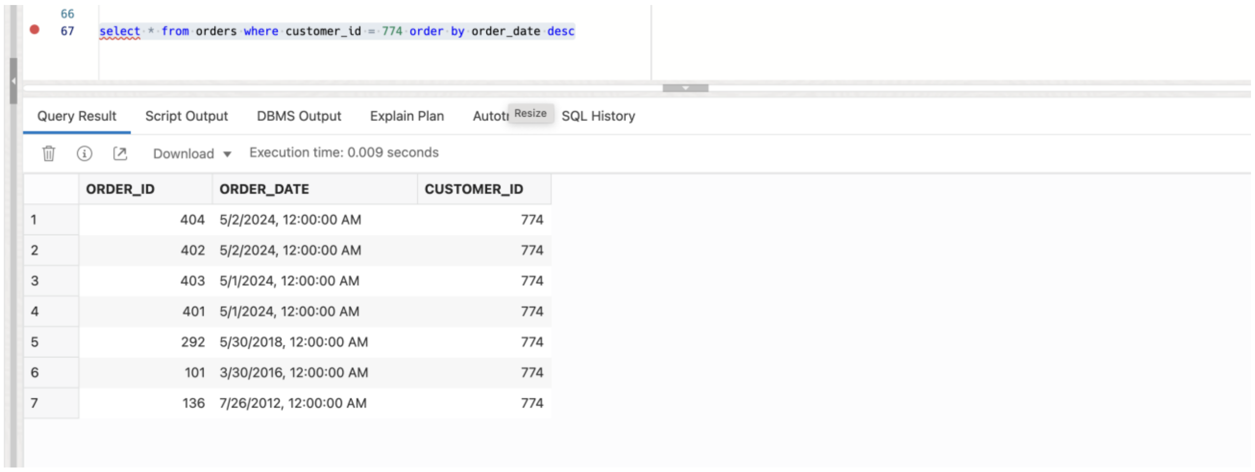


```sql
-- Query 7: Find and sort desc orders history by customer id and delete all orders from earlier than the third order date
-- In other words, delete order history and keep only the last three orders
-- BEFORE UPDATE - count how many old orders exists
select o.customer_id, count(*) as  count_orders
from orders o
where order_date = (
    select order_date
    from (
        SELECT customer_id, order_date, ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY order_date DESC) AS rn
    FROM orders)
    where rn = 3 and customer_id = o.customer_id)
group by o.customer_id
```

Execution time: 0.003 seconds

| | CUSTOMER_ID | COUNT_ORDERS |
|---|---|---|
| 1 | 774 | 2 |
| 2 | 655 | 1 |
| 3 | 665 | 2 |
| 4 | 668 | 4 |
| 5 | 796 | 1 |

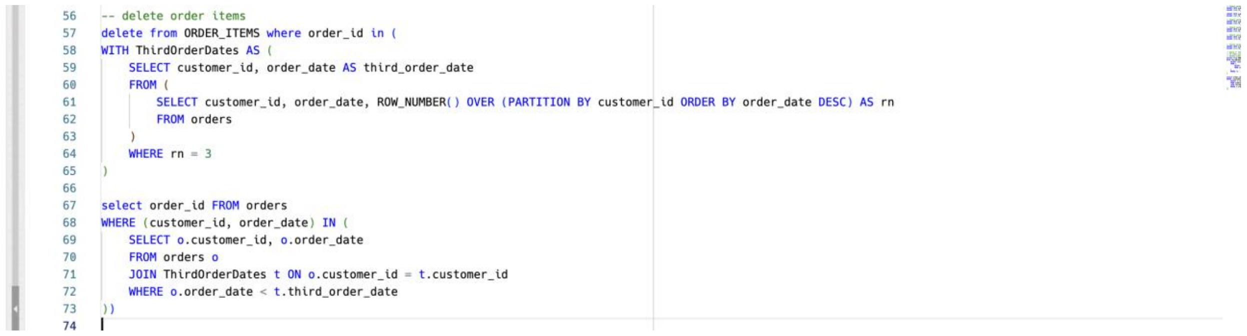
Query 7 – Customer 774 before delete:



```sql
select * from orders where customer_id = 774 order by order_date desc
```

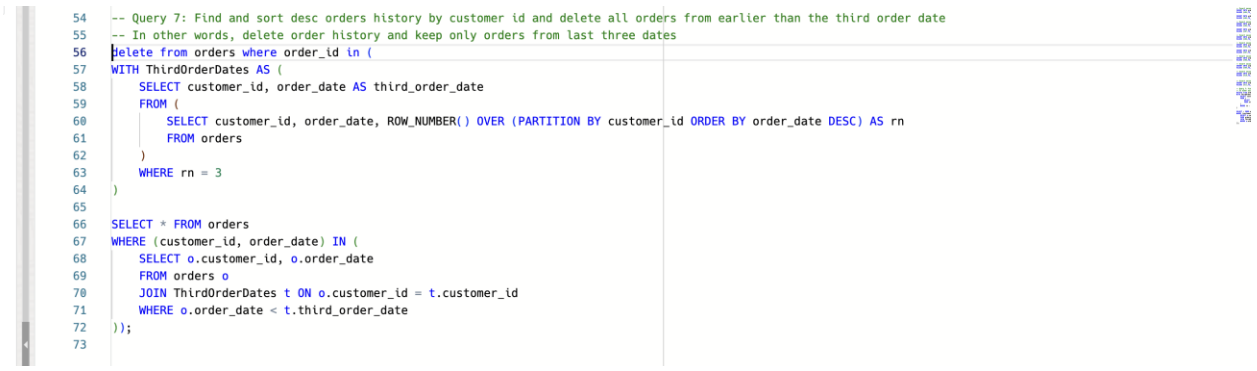
| | ORDER_ID | ORDER_DATE | CUSTOMER_ID |
|---|---|---|---|
| 1 | 404 | 5/2/2024, 12:00:00 AM | 774 |
| 2 | 402 | 5/2/2024, 12:00:00 AM | 774 |
| 3 | 403 | 5/1/2024, 12:00:00 AM | 774 |
| 4 | 401 | 5/1/2024, 12:00:00 AM | 774 |
| 5 | 292 | 5/30/2018, 12:00:00 AM | 774 |
| 6 | 101 | 3/30/2016, 12:00:00 AM | 774 |
| 7 | 136 | 7/26/2012, 12:00:00 AM | 774 |

Query 7 – Delete order items before deleting orders:



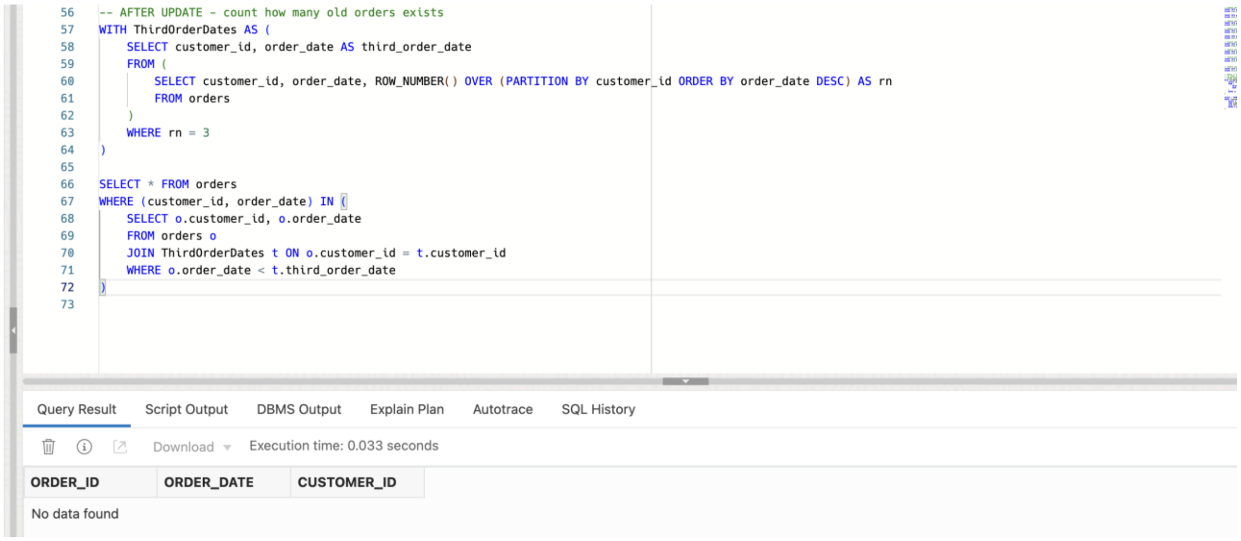
```sql
-- delete order items
delete from ORDER_ITEMS where order_id in (
WITH ThirdOrderDates AS (
    SELECT customer_id, order_date AS third_order_date
    FROM (
        SELECT customer_id, order_date, ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY order_date DESC) AS rn
        FROM orders
    )
    WHERE rn = 3
)

select order_id FROM orders
WHERE (customer_id, order_date) IN (
    SELECT o.customer_id, o.order_date
    FROM orders o
    JOIN ThirdOrderDates t ON o.customer_id = t.customer_id
    WHERE o.order_date < t.third_order_date
))
```

Query 7 – Delete:



```sql
-- Query 7: Find and sort desc orders history by customer id and delete all orders from earlier than the third order date
-- In other words, delete order history and keep only orders from last three dates
delete from orders where order_id in (
WITH ThirdOrderDates AS (
    SELECT customer_id, order_date AS third_order_date
    FROM (
        SELECT customer_id, order_date, ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY order_date DESC) AS rn
        FROM orders
    )
    WHERE rn = 3
)

SELECT * FROM orders
WHERE (customer_id, order_date) IN (
    SELECT o.customer_id, o.order_date
    FROM orders o
    JOIN ThirdOrderDates t ON o.customer_id = t.customer_id
    WHERE o.order_date < t.third_order_date
));
```

Query 7 – After delete:



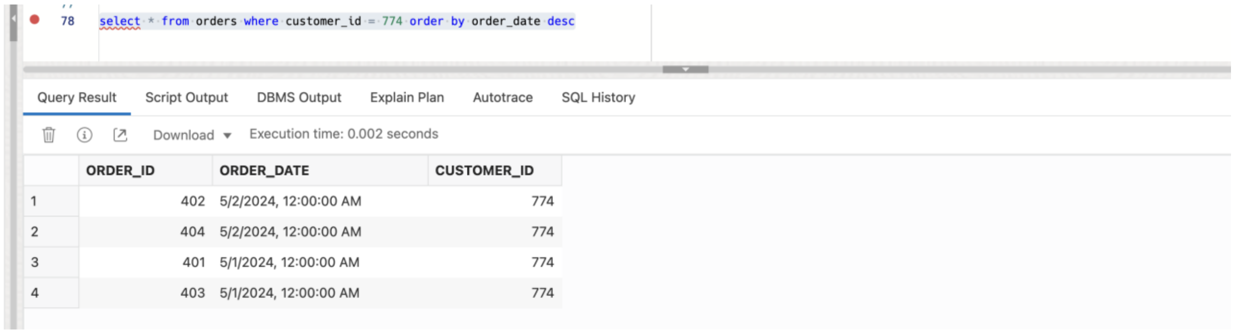
```
-- AFTER UPDATE - count how many old orders exists
WITH ThirdOrderDates AS (
    SELECT customer_id, order_date AS third_order_date
    FROM (
        SELECT customer_id, order_date, ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY order_date DESC) AS rn
        FROM orders
    )
    WHERE rn = 3
)

SELECT * FROM orders
WHERE (customer_id, order_date) IN (
    SELECT o.customer_id, o.order_date
    FROM orders o
    JOIN ThirdOrderDates t ON o.customer_id = t.customer_id
    WHERE o.order_date < t.third_order_date
)
```

Execution time: 0.033 seconds

| ORDER_ID | ORDER_DATE | CUSTOMER_ID |
|---|---|---|

No data found

Query 7 – Customer 774 after delete:

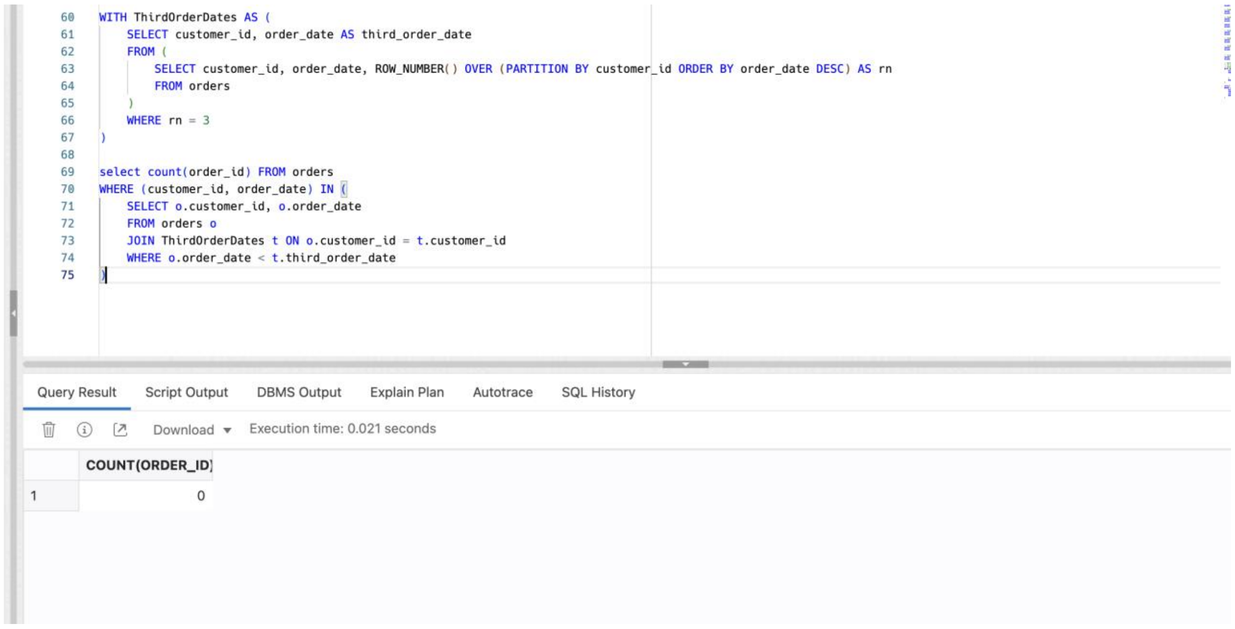


```
select * from orders where customer_id = 774 order by order_date desc
```

Execution time: 0.002 seconds

| | ORDER_ID | ORDER_DATE | CUSTOMER_ID |
|---|---|---|---|
| 1 | 402 | 5/2/2024, 12:00:00 AM | 774 |
| 2 | 404 | 5/2/2024, 12:00:00 AM | 774 |
| 3 | 401 | 5/1/2024, 12:00:00 AM | 774 |
| 4 | 403 | 5/1/2024, 12:00:00 AM | 774 |

Query 7 – Table after delete:

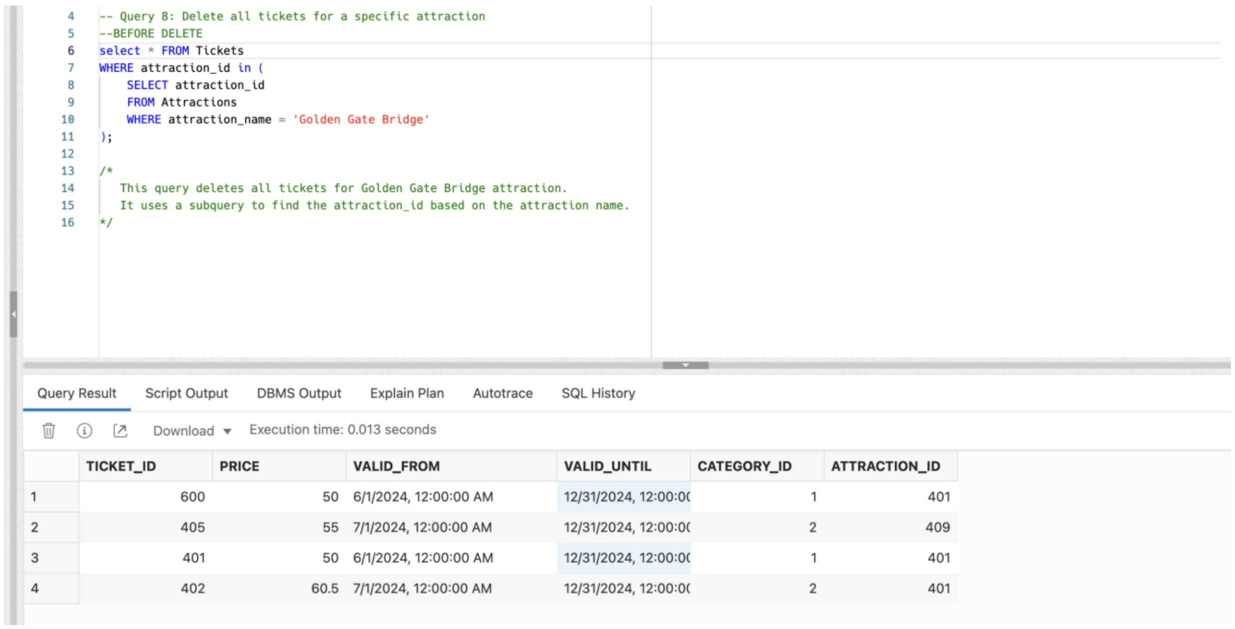


```
WITH ThirdOrderDates AS (
    SELECT customer_id, order_date AS third_order_date
    FROM (
        SELECT customer_id, order_date, ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY order_date DESC) AS rn
        FROM orders
    )
    WHERE rn = 3
)

select count(order_id) FROM orders
WHERE (customer_id, order_date) IN (
    SELECT o.customer_id, o.order_date
    FROM orders o
    JOIN ThirdOrderDates t ON o.customer_id = t.customer_id
    WHERE o.order_date < t.third_order_date
)
```

Execution time: 0.021 seconds

| | COUNT(ORDER_ID) |
|---|---|
| 1 | 0 |

Query 8 – Before delete:

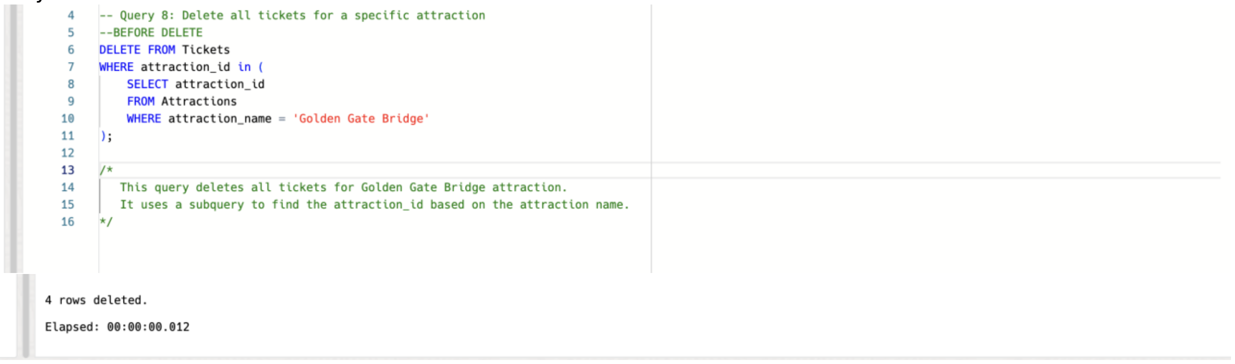


```
-- Query 8: Delete all tickets for a specific attraction
--BEFORE DELETE
select * FROM Tickets
WHERE attraction_id in (
    SELECT attraction_id
    FROM Attractions
    WHERE attraction_name = 'Golden Gate Bridge'
);

/*
   This query deletes all tickets for Golden Gate Bridge attraction.
   It uses a subquery to find the attraction_id based on the attraction name.
*/
```

Execution time: 0.013 seconds

| | TICKET_ID | PRICE | VALID_FROM | VALID_UNTIL | CATEGORY_ID | ATTRACTION_ID |
|---|---|---|---|---|---|---|
| 1 | 600 | 50 | 6/1/2024, 12:00:00 AM | 12/31/2024, 12:00:0( | 1 | 401 |
| 2 | 405 | 55 | 7/1/2024, 12:00:00 AM | 12/31/2024, 12:00:0( | 2 | 409 |
| 3 | 401 | 50 | 6/1/2024, 12:00:00 AM | 12/31/2024, 12:00:0( | 1 | 401 |
| 4 | 402 | 60.5 | 7/1/2024, 12:00:00 AM | 12/31/2024, 12:00:0( | 2 | 401 |

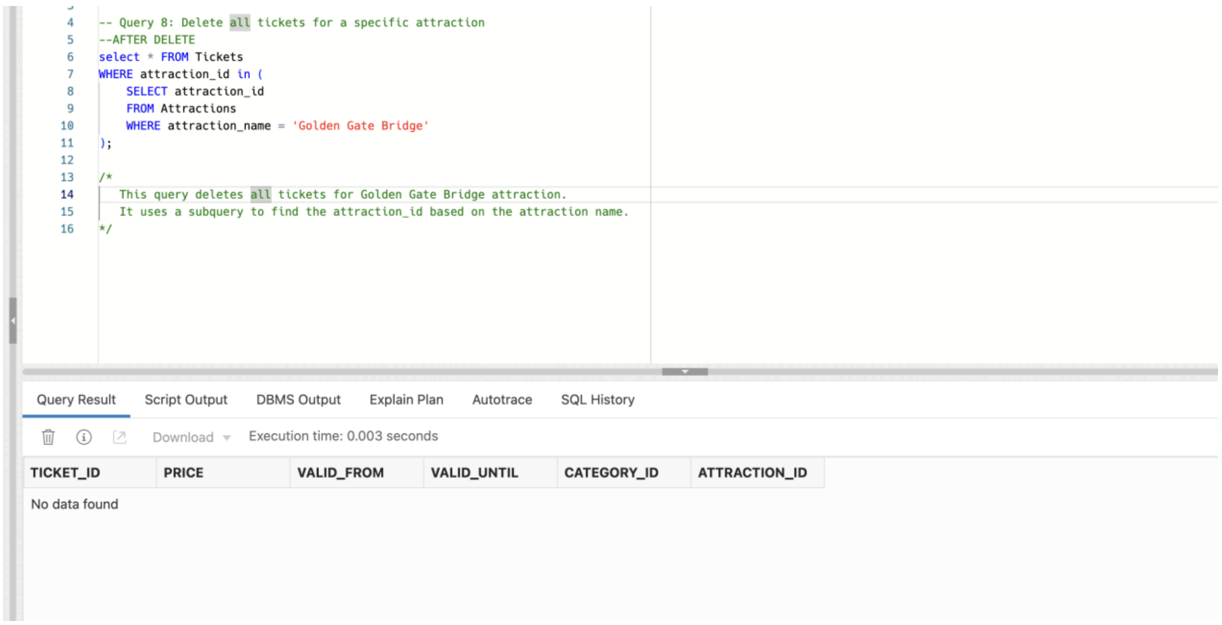
Query 8 – Delete:



```
-- Query 8: Delete all tickets for a specific attraction
--BEFORE DELETE
DELETE FROM Tickets
WHERE attraction_id in (
    SELECT attraction_id
    FROM Attractions
    WHERE attraction_name = 'Golden Gate Bridge'
);

/*
   This query deletes all tickets for Golden Gate Bridge attraction.
   It uses a subquery to find the attraction_id based on the attraction name.
*/
```

```
4 rows deleted.

Elapsed: 00:00:00.012
```

Query 8 – Table after delete:



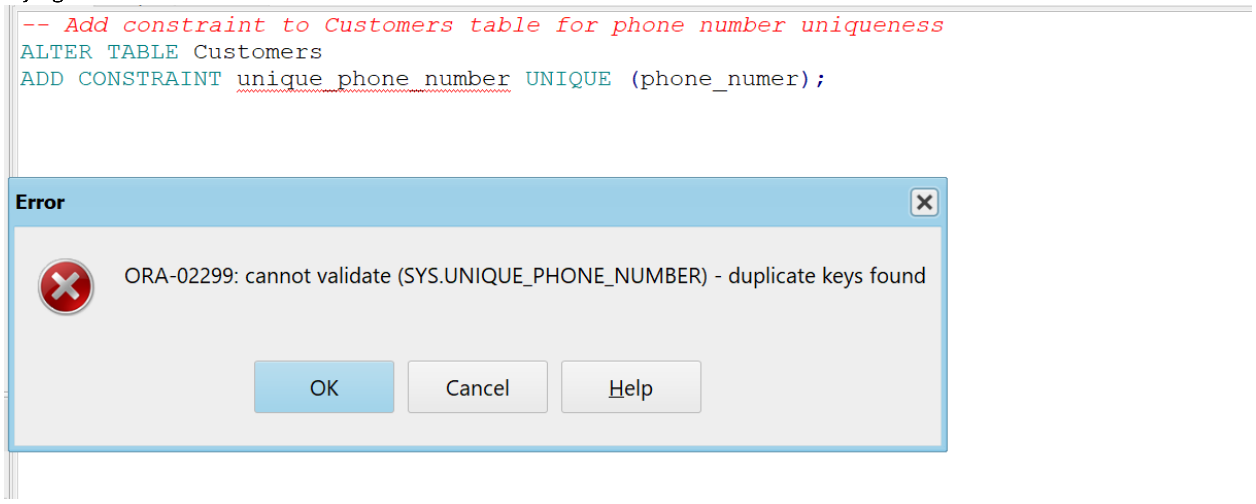
```
-- Query 8: Delete all tickets for a specific attraction
--AFTER DELETE
select * FROM Tickets
WHERE attraction_id in (
    SELECT attraction_id
    FROM Attractions
    WHERE attraction_name = 'Golden Gate Bridge'
);

/*
  This query deletes all tickets for Golden Gate Bridge attraction.
  It uses a subquery to find the attraction_id based on the attraction name.
*/
```

```
Execution time: 0.003 seconds

TICKET_ID  PRICE  VALID_FROM  VALID_UNTIL  CATEGORY_ID  ATTRACTION_ID
No data found
```

Constraints:

Constraint 1:

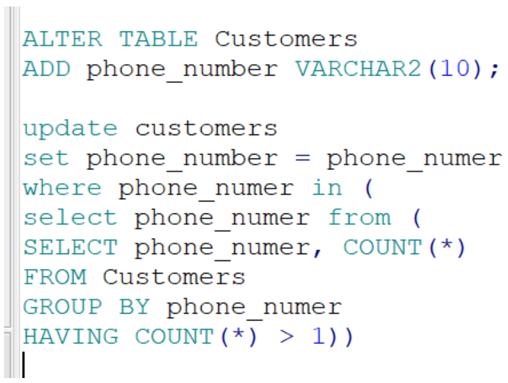
Trying to add the constraint



```
-- Add constraint to Customers table for phone number uniqueness
ALTER TABLE Customers
ADD CONSTRAINT unique_phone_number UNIQUE (phone_numer);
```

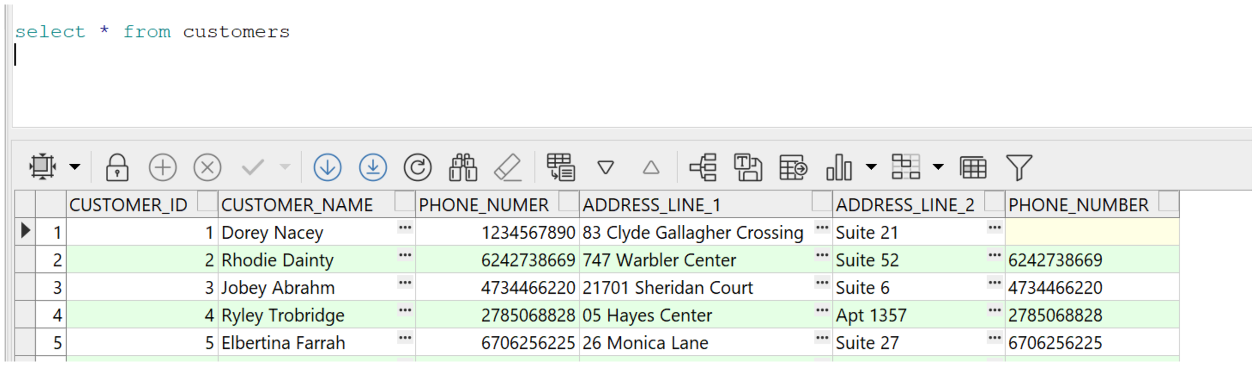
```
Error
ORA-02299: cannot validate (SYS.UNIQUE_PHONE_NUMBER) - duplicate keys found
```

Add new phone number field and copy the data



```
ALTER TABLE Customers
ADD phone_number VARCHAR2(10);

update customers
set phone_number = phone_numer
where phone_numer in (
select phone_numer from (
SELECT phone_numer, COUNT(*)
FROM Customers
GROUP BY phone_numer
HAVING COUNT(*) > 1))
```

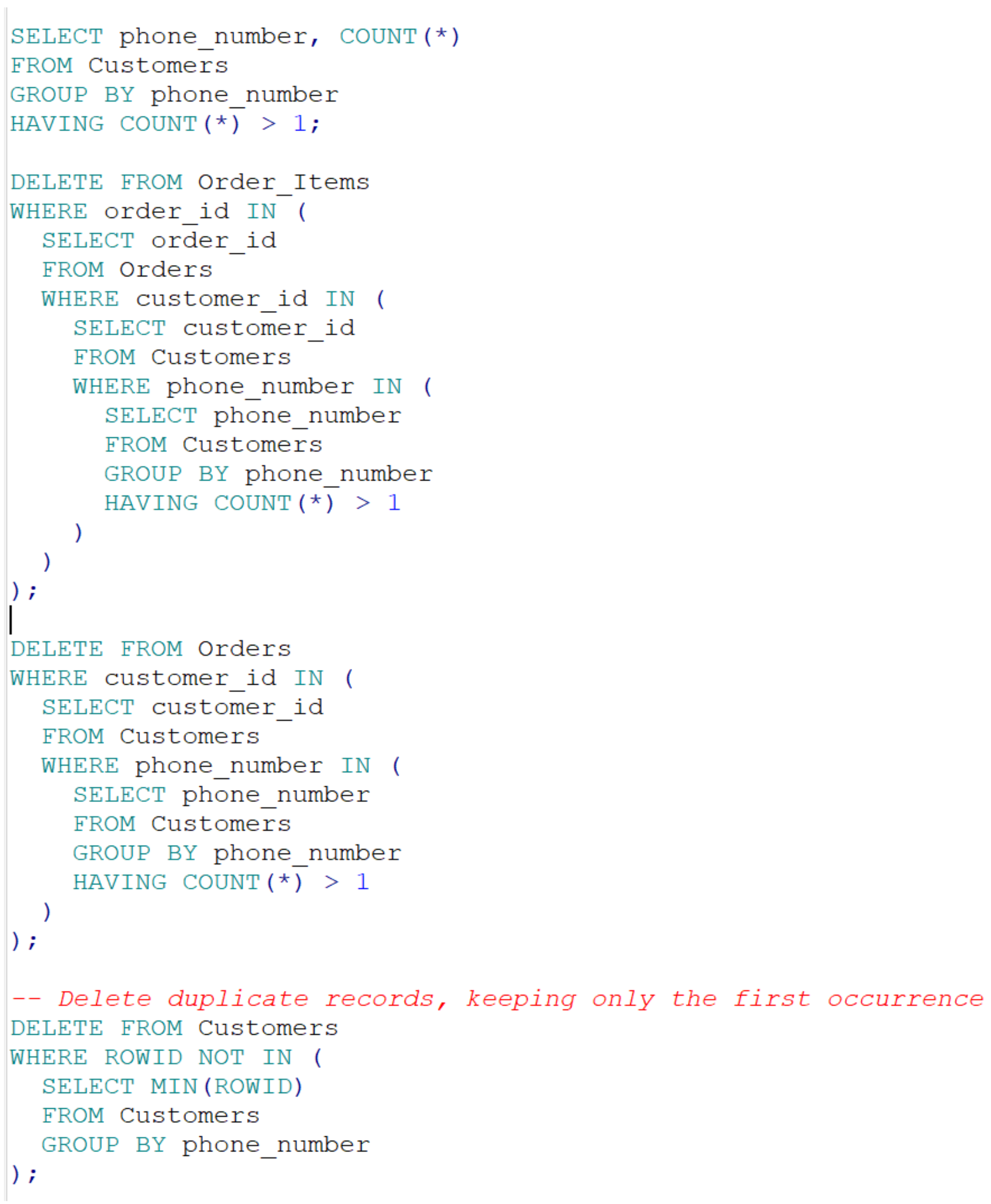
Validate the data has been copied



```
select * from customers
```

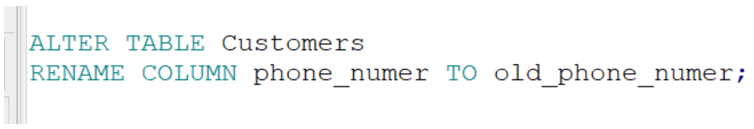
| | CUSTOMER_ID | CUSTOMER_NAME | PHONE_NUMER | ADDRESS_LINE_1 | ADDRESS_LINE_2 | PHONE_NUMBER |
|---|---|---|---|---|---|---|
| 1 | 1 | Dorey Nacey | 1234567890 | 83 Clyde Gallagher Crossing | Suite 21 | |
| 2 | 2 | Rhodie Dainty | 6242738669 | 747 Warbler Center | Suite 52 | 6242738669 |
| 3 | 3 | Jobey Abrahm | 4734466220 | 21701 Sheridan Court | Suite 6 | 4734466220 |
| 4 | 4 | Ryley Trobridge | 2785068828 | 05 Hayes Center | Apt 1357 | 2785068828 |
| 5 | 5 | Elbertina Farrah | 6706256225 | 26 Monica Lane | Suite 27 | 6706256225 |

Delete duplicates before adding the constraint



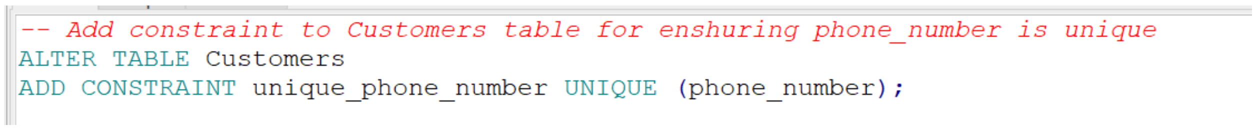
```sql
SELECT phone_number, COUNT(*)
FROM Customers
GROUP BY phone_number
HAVING COUNT(*) > 1;

DELETE FROM Order_Items
WHERE order_id IN (
  SELECT order_id
  FROM Orders
  WHERE customer_id IN (
    SELECT customer_id
    FROM Customers
    WHERE phone_number IN (
      SELECT phone_number
      FROM Customers
      GROUP BY phone_number
      HAVING COUNT(*) > 1
    )
  )
);

DELETE FROM Orders
WHERE customer_id IN (
  SELECT customer_id
  FROM Customers
  WHERE phone_number IN (
    SELECT phone_number
    FROM Customers
    GROUP BY phone_number
    HAVING COUNT(*) > 1
  )
);

-- Delete duplicate records, keeping only the first occurrence
DELETE FROM Customers
WHERE ROWID NOT IN (
  SELECT MIN(ROWID)
  FROM Customers
  GROUP BY phone_number
);
```

Rename old field



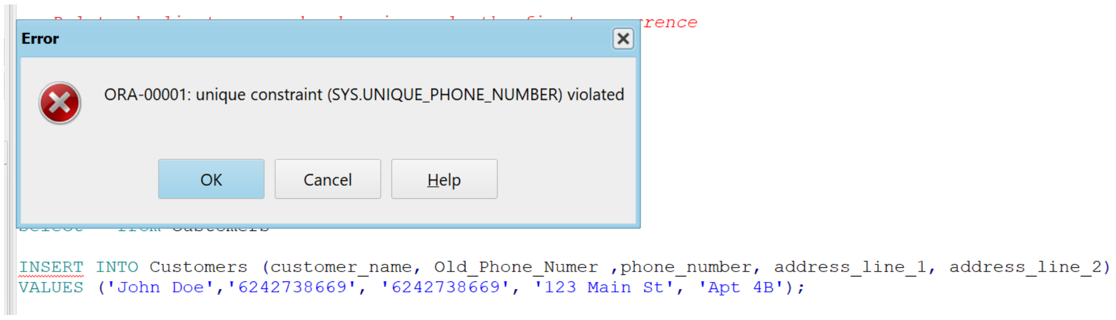
```sql
ALTER TABLE Customers
RENAME COLUMN phone_numer TO old_phone_numer;
```

Adding the constraint



```sql
-- Add constraint to Customers table for enshuring phone_number is unique
ALTER TABLE Customers
ADD CONSTRAINT unique_phone_number UNIQUE (phone_number);
```

Attempt to insert invalid data

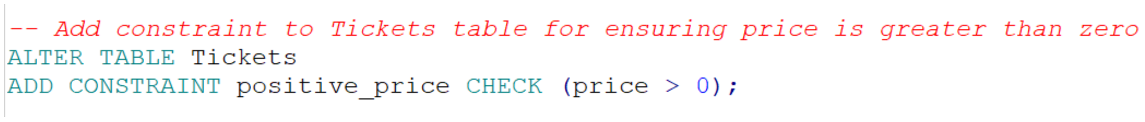


```
ORA-00001: unique constraint (SYS.UNIQUE_PHONE_NUMBER) violated
```

```
INSERT INTO Customers (customer_name, Old_Phone_Numer ,phone_number, address_line_1, address_line_2)
VALUES ('John Doe','6242738669', '6242738669', '123 Main St', 'Apt 4B');
```

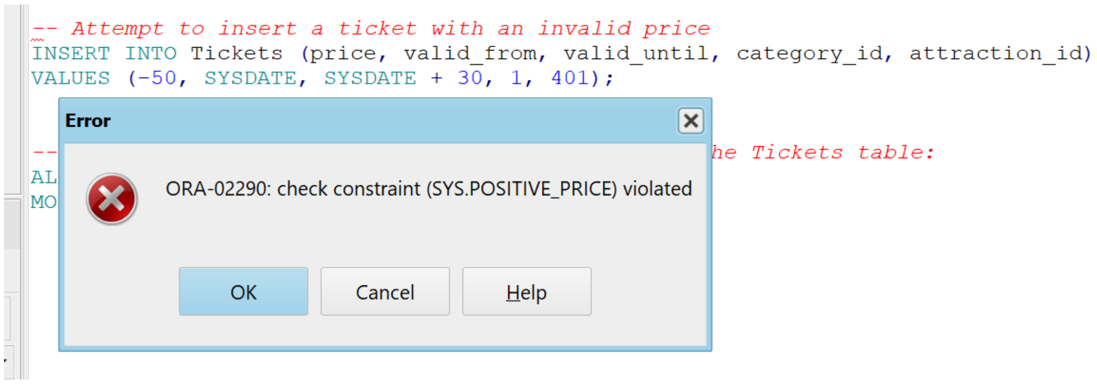
Constraint 2:

Adding the constraint



```
-- Add constraint to Tickets table for ensuring price is greater than zero
ALTER TABLE Tickets
ADD CONSTRAINT positive_price CHECK (price > 0);
```

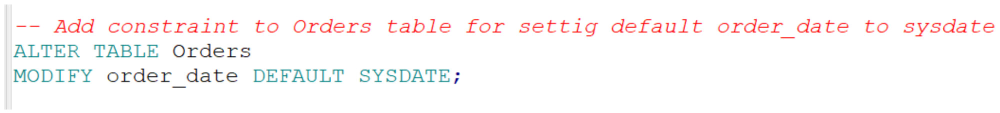
Attempt to insert invalid data



```
-- Attempt to insert a ticket with an invalid price
INSERT INTO Tickets (price, valid_from, valid_until, category_id, attraction_id)
VALUES (-50, SYSDATE, SYSDATE + 30, 1, 401);
```

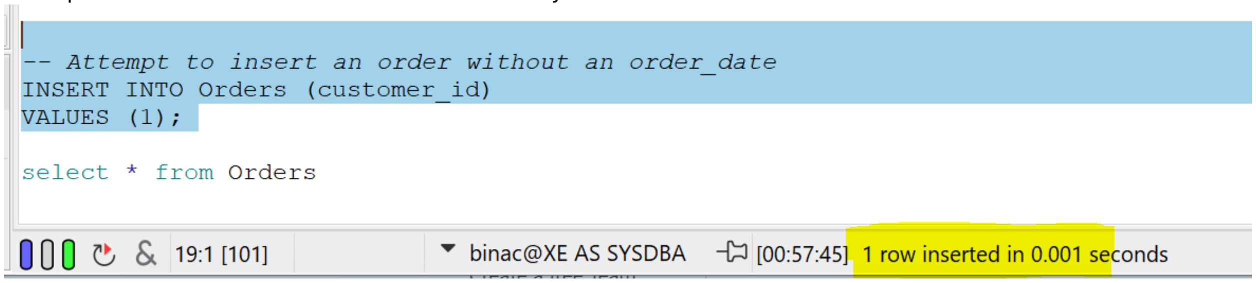
```
ORA-02290: check constraint (SYS.POSITIVE_PRICE) violated
```

Constraint 3:



```
-- Add constraint to Orders table for settig default order_date to sysdate
ALTER TABLE Orders
MODIFY order_date DEFAULT SYSDATE;
```

Attempt to insert the a line without the new default key



```
-- Attempt to insert an order without an order_date
INSERT INTO Orders (customer_id)
VALUES (1);

select * from Orders
```

```
19:1 [101]    binac@XE AS SYSDBA    [00:57:45]  1 row inserted in 0.001 seconds
```