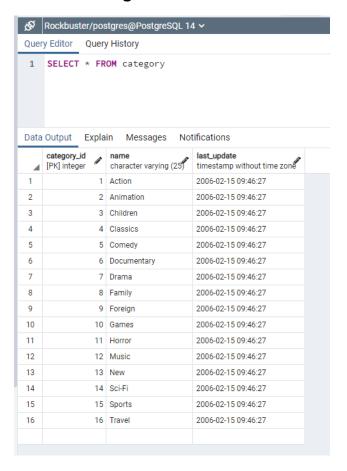
Rockbuster Stealth PostgreSQL queries:

Contents:

- 1. Selecting
- 2. Inserting rows
- 3. Updating and deleting
- 4. Creating tables
- 5. Ordering
- 6. Aggregating and Grouping
- 7. Using 'CASE'
- 8. Checking for duplicates
- 9. Viewing unique values
- 10. Checking for missing data
- 11. Using inner joins
- 12. Using inner joins with 'WHERE'
- 13. Using Subqueries
- 14. Using Common Table Expressions

1. Selecting



2. Inserting rows

```
INSERT INTO category(category_id,name,last_update)
VALUES(21,'War',current_timestamp)

INSERT INTO category(category_id,name,last_update)
VALUES(20,'Romance',current_timestamp)

INSERT INTO category(category_id,name,last_update)
VALUES(19,'Mystery',current_timestamp)

INSERT INTO category(category_id,name,last_update)
VALUES(18,'Crime',current_timestamp)

INSERT INTO category(category_id,name,last_update)
VALUES(17,'Thriller',current_timestamp)
```

3. Updating and deleting

```
1  UPDATE film_category
2  SET category_id = 17
3  WHERE film_id = 5

1  DELETE FROM category
2  WHERE category_id = 19
```

4. Creating Tables

```
Query Editor Query History
    CREATE TABLE employees
1
2
3
         employee_id serial NOT NULL,
4
         name VARCHAR(50),
5
         contact_number INT,
         designation_id INT,
6
         last_update TIMESTAMP NOT NULL DEFAULT NOW(),
7
         CONSTRAINT employees_pkey PRIMARY KEY (employee_id)
8
9
   );
Data Output
            Explain
                     Messages
                                 Notifications
   employee_id
                                    contact_number
                                                      designation_id
                                                                      last_update
                                                                      timestamp without time zone
[PK] integer
                  character varying (50)
                                    integer
                                                      integer
```

5. Ordering

```
1 SELECT * FROM film
2 ORDER BY
3 title,
4 release_year DESC,
5 rental_rate DESC;
6
```

6. Aggregating and Grouping

```
SELECT rating,
 1
              AVG(rental_rate) AS avg_rental_rate,
 2
              MIN(rental_duration) AS min_rental_duration,
 3
              MAX(rental_duration) AS max_rental_duration
 4
     FROM film
 5
    GROUP BY rating
 6
 7
Data Output
             Explain Messages
                                  Notifications
                                                      max_rental_duration_
                                    min_rental_duration_.
                avg_rental_rate
                                    smallint
                                                       smallint
   mpaa_rating
                numeric
1
   R
                  2.9387179487179487
                                                    3
                                                                       7
  NC-17
                   2.970952380952381
                                                    3
                                                                       7
2
                                                                       7
3
   G
                   2.888876404494382
                                                    3
   PG
                  3.0518556701030928
                                                    3
                                                                       7
4
5
   PG-13
                   3.034843049327354
                                                    3
                                                                       7
```

7. Using 'CASE'

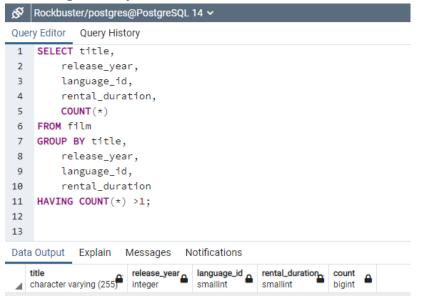
5 NC-17

```
MIN(replacement_cost) AS min_replacement_cost,
            MAX(replacement_cost) AS max_replacement_cost
4 FROM film
5 GROUP BY rating
6 ORDER BY CASE WHEN rating = 'G' THEN 1
                     WHEN rating = 'PG' THEN 2
7
8
                     WHEN rating = 'PG-13' THEN 3
9
                     WHEN rating = 'R' THEN 4
                     WHEN rating = 'NC-17' THEN 5
10
11
                 END;
Data Output Explain Messages Notifications
rating min_replacement_cost numeric max_replacement_cost numeric
1 G
                           9.99
                                             29.99
3 PG-13
                           9.99
                                             29.99
4 R
                           9.99
                                             29.99
```

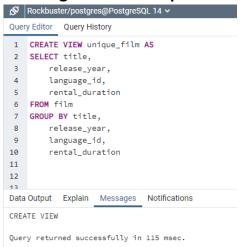
9.99

29.99

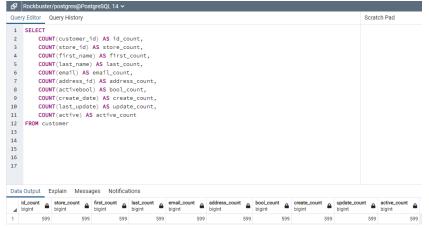
8. Checking for duplicate records



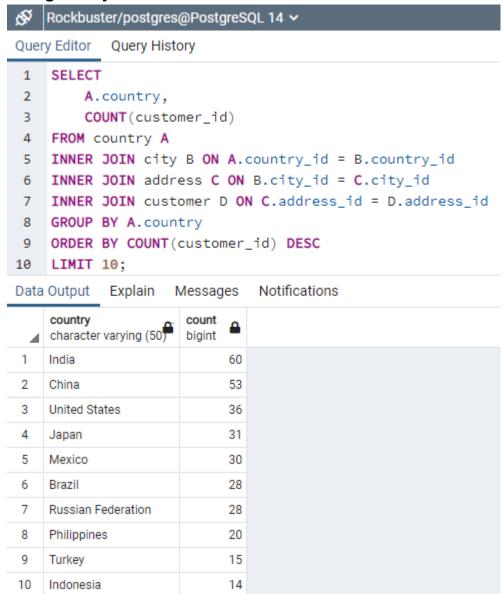
9. Creating view for unique values



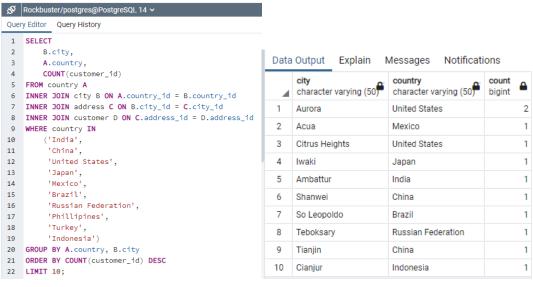
10. Checking for missing data



11. Using inner joins



12. Using inner joins with 'WHERE'



13. Using subqueries

```
1 SELECT
        country.country,
 2
 3
         COUNT(DISTINCT customer.customer_id) AS all_customer_count,
         COUNT(DISTINCT top_5.customer_id) AS top_customer_count
 4
 6 INNER JOIN city ON country.country_id = city.country_id
 7
    INNER JOIN address ON city.city_id = address.city_id
    INNER JOIN customer ON address.address_id = customer.address_id
 8
 9 LEFT JOIN
10
        (SELECT
            E.customer_id,
11
12
            D.first_name,
13
            D.last_name,
14
            A.country,
15
            B.city,
             SUM(E.amount) AS total_paid
16
17
         FROM country A
18
         INNER JOIN city B ON A.country_id = B.country_id
         INNER JOIN address C ON B.city_id = C.city_id
19
        INNER JOIN customer D ON C.address_id = D.address_id
20
        INNER JOIN payment E ON D.customer_id = E.customer_id
21
22
        WHERE country IN
           ('India','China','United States','Japan','Mexico',
23
            'Brazil', 'Russian Federation', 'Phillipines', 'Turkey', 'Indonesia')
24
25
           AND city IN
26
            ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambatttur', 'Shanwei',
27
            'So Leopoldo', 'Teboksary', 'Tianjin', 'Cianjur')
28
       GROUP BY
29
          E.customer_id,
30
           D.first_name,
31
          D.last_name,
32
          A.country,
33
           B.city
      ORDER BY total_paid DESC
34
35
       LIMIT 5) AS top_5 ON top_5.country = country.country
36 GROUP BY country.country
37 ORDER BY top_customer_count DESC
Data Output Explain Messages Notifications
                               all_customer_count
                                             top_customer_count
  biaint
                                              bigint
    United States
                                           36
                                                          2
 2 Mexico
                                           30
                                                          1
 3 Japan
                                           31
                                                          1
 4 China
                                           53
                                                          1
                                            1
                                                          0
 5 Anguilla
```

14. Using Common Table Expressions

```
Rockbuster/postgres@PostgreSQL 14 v
Query Editor Query History
    WITH top_5_cte (amount) AS
 1
 2
         (SELECT
 3
            E.customer_id,
 4
             D.first_name,
            D.last_name,
 5
 6
            A.country,
 7
             B.city,
             SUM(E.amount) AS total_paid
 8
 9
         FROM country A
         INNER JOIN city B ON A.country_id = B.country_id
10
         INNER JOIN address C ON B.city_id = C.city_id
11
12
         INNER JOIN customer D ON C.address_id = D.address_id
13
         INNER JOIN payment E ON D.customer_id = E.customer_id
14
         WHERE country IN
             ('India', 'China', 'United States', 'Japan', 'Mexico',
15
              'Brazil', 'Russian Federation', 'Phillipines', 'Turkey', 'Indonesia')
16
             AND city IN
17
18
             ('Aurora', 'Acua', 'Citrus Heights', 'Iwaki', 'Ambatttur', 'Shanwei',
              'So Leopoldo', 'Teboksary', 'Tianjin', 'Cianjur')
19
20
        GROUP BY
21
            E.customer_id,
             D.first_name,
22
23
             D.last_name,
24
             A.country,
25
             B.city
         ORDER BY total_paid DESC
26
        LIMIT 5)
27
28 SELECT AVG(total_paid) AS average_top_5
29 FROM top_5_cte
Data Output Explain Messages Notifications
   average_top_5
 102.55600000000000000
```