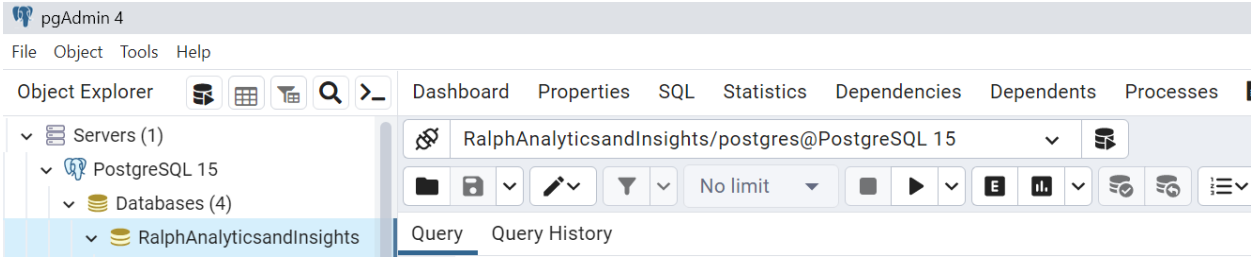
SQL POST

**SQL--Project Scenarios in The Real World**

Hello Analyst, at this time, allow me to show you my real project scenarios. I solved analytical problems using SQL. I used PostgreSQL, by the way.




A little bit introduction and reminder about SQL

"Structured query language (SQL) is a programming language for storing and processing information in a relational database. A relational database stores information in tabular form, with rows and columns representing different data attributes and the various relationships between the data values. You can use SQL statements to store, update, remove, search, and retrieve information from the database. You can also use SQL to maintain and optimize database performance."

SELECT, FROM, COUNT, UNIQUE, DISTINCT, CASE-WHEN, LIKE, ALIASES, CONCATENATE, ORDER BY, GROUP BY, JOINS, AGGREGATE FUNCTIONS (SUM, AVG, etc.), DATE/TIME, EXTRACT, SUB QUERY (CORRELATED AND UNCORRELATED)

Stick close and enjoy and find useful comments in between query lines.

SOME KEY TERMS:

- A Primary Key (PK) is used to ensure that data in the specific column is unique. A column cannot have NULL values. It is either an existing table column or a column that is specifically generated by the database according to a defined sequence.
- A Foreign Key (FK) is a column or group of columns in a relational database table that provides a link between data in two tables. It is a column (or columns) that references a column (most often the primary key) of another table.
- A JOIN clause is used to combine rows from two or more tables, based on a related column between them.
- The INNER JOIN keyword selects records that have matching values in both tables.
- The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.
- The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.
- The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country". The GROUP BY statement is often used with aggregate functions ( COUNT (), MAX (), MIN (), SUM (), AVG () ) to group the result-set by one or more columns.
- The CASE expression goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause. If there is no ELSE part and no conditions are true, it returns NULL.
- The subquery is a nested query.
- When this subquery is executed only once and the result of this subquery is used to extract the data in the main query, then this type of subquery is known as UNCORRELATED subquery.
- when a subquery refers to the main query for each execution, then the subquery is known as CORRELATED subquery.

/*

For this project, several tables were used ant the top 10 rows of each tables and their respective column names and attributes are displayed to give a feel of what each query line entails:

-- Also, find attached a picture of what the data output from final queries look like.

i. film_table of 1000 rows with columns film_id, title, length, replacement_cost, and rating

ii. film_category_table of 1000 rows with columns film_id, category_id

iii. category_table of 16 rows with columns category_id, name

iv. film_actor_table of 5462 rows with columns actor_id, film_id

v. actor_table of 200 rows with columns actor_id, first_name, last_name

vi. customer_table of 599 rows with columns customer_id, store_id, first_name, last_name, email, address_id

vii. address_table of 603 rows with columns address_id, district, city_id, postal_code, phone

viii. city_table of 600 rows with columns city_id, city, country_id

ix. payment_table of 16049 rows with columns payment_id, customer_id, staff_id, rental_id, amount

x. inventory_table of 4581 rows with columns inventory_id, film_id, store_id

xi. rental_table of 16044 rows with columns rental_id, rental_date, inventory_id, customer_id, return_date, staff_id

*/

TABLE SNIPS USED IN THIS PROJECT ARE SHOWN BELOW:

1. film_table

Data Output Messages Notifications

| | film_id [PK] integer | title text | length smallint | replacement_cost numeric (5,2) | rating mpaa_rating |
|---|---|---|---|---|---|
| 1 | 1 | ACADEMY DINOSAUR | 86 | 20.99 | PG |
| 2 | 2 | ACE GOLDFINGER | 48 | 12.99 | G |
| 3 | 3 | ADAPTATION HOLES | 50 | 18.99 | NC-17 |
| 4 | 4 | AFFAIR PREJUDICE | 117 | 26.99 | G |
| 5 | 5 | AFRICAN EGG | 130 | 22.99 | G |
| 6 | 6 | AGENT TRUMAN | 169 | 17.99 | PG |
| 7 | 7 | AIRPLANE SIERRA | 62 | 28.99 | PG-13 |
| 8 | 8 | AIRPORT POLLOCK | 54 | 15.99 | R |
| 9 | 9 | ALABAMA DEVIL | 114 | 21.99 | PG-13 |
| 10 | 10 | ALADDIN CALENDAR | 63 | 24.99 | NC-17 |

2. film_category_table

Data Output Messages Notificati

| | film_id [PK] smallint | category_id [PK] smallint |
|---|---|---|
| 1 | 1 | 6 |
| 2 | 2 | 11 |
| 3 | 3 | 6 |
| 4 | 4 | 11 |
| 5 | 5 | 8 |
| 6 | 6 | 9 |
| 7 | 7 | 5 |
| 8 | 8 | 11 |
| 9 | 9 | 11 |
| 10 | 10 | 15 |

3. category_table

Data Output Messages Notifica

| | category_id [PK] integer | name text |
|---|---|---|
| 1 | 1 | Action |
| 2 | 2 | Animation |
| 3 | 3 | Children |
| 4 | 4 | Classics |
| 5 | 5 | Comedy |
| 6 | 6 | Documentary |
| 7 | 7 | Drama |
| 8 | 8 | Family |
| 9 | 9 | Foreign |
| 10 | 10 | Games |

4. film_actor_table

Data Output Messages Notificati

| | actor_id [PK] smallint | film_id [PK] smallint |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 23 |
| 3 | 1 | 25 |
| 4 | 1 | 106 |
| 5 | 1 | 140 |
| 6 | 1 | 166 |
| 7 | 1 | 277 |
| 8 | 1 | 361 |
| 9 | 1 | 438 |
| 10 | 1 | 499 |

5. actor_table

Data Output Messages Notifications

| | actor_id [PK] integer | first_name text | last_name text |
|---|---|---|---|
| 1 | 1 | PENELOPE | GUINESS |
| 2 | 2 | NICK | WAHLBERG |
| 3 | 3 | ED | CHASE |
| 4 | 4 | JENNIFER | DAVIS |
| 5 | 5 | JOHNNY | LOLLOBRIGIDA |
| 6 | 6 | BETTE | NICHOLSON |
| 7 | 7 | GRACE | MOSTEL |
| 8 | 8 | MATTHEW | JOHANSSON |
| 9 | 9 | JOE | SWANK |
| 10 | 10 | CHRISTIAN | GABLE |

6. customer_table

Data Output Messages Notifications

| | customer_id [PK] integer | store_id smallint | first_name text | last_name text | email text | address_id smallint |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | MARY | SMITH | MARY.SMITH@sakilacustomer.org | 5 |
| 2 | 2 | 1 | PATRICIA | JOHNSON | PATRICIA.JOHNSON@sakilacustomer.org | 6 |
| 3 | 3 | 1 | LINDA | WILLIAMS | LINDA.WILLIAMS@sakilacustomer.org | 7 |
| 4 | 4 | 2 | BARBARA | JONES | BARBARA.JONES@sakilacustomer.org | 8 |
| 5 | 5 | 1 | ELIZABETH | BROWN | ELIZABETH.BROWN@sakilacustomer.org | 9 |
| 6 | 6 | 2 | JENNIFER | DAVIS | JENNIFER.DAVIS@sakilacustomer.org | 10 |
| 7 | 7 | 1 | MARIA | MILLER | MARIA.MILLER@sakilacustomer.org | 11 |
| 8 | 8 | 2 | SUSAN | WILSON | SUSAN.WILSON@sakilacustomer.org | 12 |
| 9 | 9 | 2 | MARGARET | MOORE | MARGARET.MOORE@sakilacustomer.org | 13 |
| 10 | 10 | 1 | DOROTHY | TAYLOR | DOROTHY.TAYLOR@sakilacustomer.org | 14 |

7. address_table

Data Output Messages Notifications

| | address_id [PK] integer | address text | address2 text | district text | city_id smallint | postal_code text | phone text |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 47 MySakila Drive | [null] | Alberta | 300 | | |
| 2 | 2 | 28 MySQL Boulevard | [null] | QLD | 576 | | |
| 3 | 3 | 23 Workhaven Lane | [null] | Alberta | 300 | | 14033335568 |
| 4 | 4 | 1411 Lillydale Drive | [null] | QLD | 576 | | 6172235589 |
| 5 | 5 | 1913 Hanoi Way | | Nagasaki | 463 | 35200 | 28303384290 |
| 6 | 6 | 1121 Loja Avenue | | California | 449 | 17886 | 838635286649 |
| 7 | 7 | 692 Joliet Street | | Attika | 38 | 83579 | 448477190408 |
| 8 | 8 | 1566 Inegl Manor | | Mandalay | 349 | 53561 | 705814003527 |
| 9 | 9 | 53 Idfu Parkway | | Nantou | 361 | 42399 | 10655648674 |
| 10 | 10 | 1795 Santiago de Compostela Way | | Texas | 295 | 18743 | 860452626434 |

8. city_table

Data Output Messages Notifications

| | city_id [PK] integer | city text | country_id smallint |
|---|---|---|---|
| 1 | 1 | A Corua (La Corua) | 87 |
| 2 | 2 | Abha | 82 |
| 3 | 3 | Abu Dhabi | 101 |
| 4 | 4 | Acua | 60 |
| 5 | 5 | Adana | 97 |
| 6 | 6 | Addis Abeba | 31 |
| 7 | 7 | Aden | 107 |
| 8 | 8 | Adoni | 44 |
| 9 | 9 | Ahmadnagar | 44 |
| 10 | 10 | Akishima | 50 |

9. payment_table

Data Output Messages Notifications

| | payment_id integer | customer_id smallint | staff_id smallint | rental_id integer | amount numeric (5,2) |
|---|---|---|---|---|---|
| 1 | 16050 | 269 | 2 | 7 | 1.99 |
| 2 | 16051 | 269 | 1 | 98 | 0.99 |
| 3 | 16052 | 269 | 2 | 678 | 6.99 |
| 4 | 16053 | 269 | 2 | 703 | 0.99 |
| 5 | 16054 | 269 | 1 | 750 | 4.99 |
| 6 | 16055 | 269 | 2 | 1099 | 2.99 |
| 7 | 16056 | 270 | 1 | 193 | 1.99 |
| 8 | 16057 | 270 | 1 | 1040 | 4.99 |
| 9 | 16058 | 271 | 1 | 1096 | 8.99 |
| 10 | 16059 | 272 | 1 | 33 | 0.99 |

10. inventory_table

Data Output Messages Notifications

| | inventory_id [PK] integer | film_id smallint | store_id smallint |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 2 | 2 | 1 | 1 |
| 3 | 3 | 1 | 1 |
| 4 | 4 | 1 | 1 |
| 5 | 5 | 1 | 2 |
| 6 | 6 | 1 | 2 |
| 7 | 7 | 1 | 2 |
| 8 | 8 | 1 | 2 |
| 9 | 9 | 2 | 2 |
| 10 | 10 | 2 | 2 |

11. rental_table

Data Output Messages Notifications

| | rental_id [PK] integer | rental_date timestamp with time zone | inventory_id integer | customer_id smallint | return_date timestamp with time zone | staff_id smallint |
|---|---|---|---|---|---|---|
| 1 | 2 | 2005-05-24 22:54:33+01 | 1525 | 459 | 2005-05-28 19:40:33+01 | 1 |
| 2 | 3 | 2005-05-24 23:03:39+01 | 1711 | 408 | 2005-06-01 22:12:39+01 | 1 |
| 3 | 4 | 2005-05-24 23:04:41+01 | 2452 | 333 | 2005-06-03 01:43:41+01 | 2 |
| 4 | 5 | 2005-05-24 23:05:21+01 | 2079 | 222 | 2005-06-02 04:33:21+01 | 1 |
| 5 | 6 | 2005-05-24 23:08:07+01 | 2792 | 549 | 2005-05-27 01:32:07+01 | 1 |
| 6 | 7 | 2005-05-24 23:11:53+01 | 3995 | 269 | 2005-05-29 20:34:53+01 | 2 |
| 7 | 8 | 2005-05-24 23:31:46+01 | 2346 | 239 | 2005-05-27 23:33:46+01 | 2 |
| 8 | 9 | 2005-05-25 00:00:40+01 | 2580 | 126 | 2005-05-28 00:22:40+01 | 1 |
| 9 | 10 | 2005-05-25 00:02:21+01 | 1824 | 399 | 2005-05-31 22:44:21+01 | 2 |
| 10 | 11 | 2005-05-25 00:09:02+01 | 4443 | 142 | 2005-06-02 20:56:02+01 | 2 |

# QUESTIONS AND DATA QUERY OUTPUT

## QUESTION 1

Query  Query History

```
-- 1. Create a list of all the different (distinct) replacement costs of the films
select * from film

select distinct (replacement_cost)
from film

-- Question: What's the lowest replacement cost?
select min(distinct (replacement_cost))
from film
-- Answer: 9.99  (See data output snip)
-- Here, 'DISTINCT' Query was used
```

ANSWER 1

Query  Query History

```
-- 1. Create a list of all the different (distinct) replacement costs of the films
select * from film

select distinct (replacement_cost)
from film

-- Question: What's the lowest replacement cost?
select min(distinct (replacement_cost))
from film
```

Data Output  Messages  Notifications

| | replacement_cost numeric (5,2) |
|---|---|
| 1 | 19.99 |
| 2 | 25.99 |
| 3 | 13.99 |
| 4 | 10.99 |
| 5 | 23.99 |

Data Output  Messages  Notifications

| | min numeric |
|---|---|
| 1 | 9.99 |

# QUESTION 2

Query  Query History

```
/* 2. Write a query that gives an overview of how many films
       have replacements costs in the following cost ranges
        low: 9.99 - 19.99
        medium: 20.00 - 24.99
        high: 25.00 - 29.99             */

select * from film

select film_id, replacement_cost,
case when replacement_cost <= 19.99 Then 'low'
when replacement_cost BETWEEN 20 and 24.99 Then 'medium'
when replacement_cost BETWEEN 24.99 and 29.99 Then 'high'
End as cost_category
from film

-- Question: How many films have a replacement cost in the "low" group?
select count('low'),
case when replacement_cost <= 19.99 Then 'low'
when replacement_cost BETWEEN 20 and 24.99 Then 'medium'
when replacement_cost BETWEEN 24.99 and 29.99 Then 'high'
End as cost_category
from film
Group by cost_category
```

ANSWER 2

Query  Query History

```

select film_id, replacement_cost,
case when replacement_cost <= 19.99 Then 'low'
when replacement_cost BETWEEN 20 and 24.99 Then 'medium'
when replacement_cost BETWEEN 24.99 and 29.99 Then 'high'
End as cost_category
from film

-- Question: How many films have a replacement cost in the "low" group?
select count('low'),
case when replacement_cost <= 19.99 Then 'low'
when replacement_cost BETWEEN 20 and 24.99 Then 'medium'
```

Data Output  Messages  Notifications

| | film_id [PK] integer | replacement_cost numeric (5,2) | cost_category text |
|---|---|---|---|
| 1 | 1 | 20.99 | medium |
| 2 | 2 | 12.99 | low |
| 3 | 3 | 18.99 | low |
| 4 | 4 | 26.99 | high |
| 5 | 5 | 22.99 | medium |

Query  Query History

```sql
-- Question: How many films have a replacement cost in the "low" group?
select count('low'),
case when replacement_cost <= 19.99 Then 'low'
when replacement_cost BETWEEN 20 and 24.99 Then 'medium'
when replacement_cost BETWEEN 24.99 and 29.99 Then 'high'
End as cost_category
from film
Group by cost_category


/* 3. Create a list of the film titles including their title, length, and category nam
```

Data Output  Messages  Notifications

| | count bigint | cost_category text |
|---|---|---|
| 1 | 250 | medium |
| 2 | 236 | high |
| 3 | 514 | low |

## QUESTION 3

Query  Query History

```sql
/* 3. Create a list of the film titles including their title, length, and category name
      ordered descendingly by length.
      Filter the results to only the movies in the category 'Drama' or 'Sports'  */

select * from film_category
select * from film
select * from category

select title, length, name
from
film fm inner join film_category fc
on fm.film_id = fc.film_id
inner join category ct
on fc.category_id = ct.category_id
where name LIKE '%Drama%' OR name LIKE '%Sports%'

-- Question: In which category is the longest film and how long is it?
```

ANSWER 3

Query Query History

```
select * from category

select title, length, name
from
film fm inner join film_category fc
on fm.film_id = fc.film_id
inner join category ct
on fc.category_id = ct.category_id
where name LIKE '%Drama%' OR name LIKE '%Sports%'

-- Question: In which category is the longest film and how long is it?
select title, length, name
```

Data Output Messages Notifications

| | title text | length smallint | name text |
|---|---|---|---|
| 1 | ALADDIN CALENDAR | 63 | Sports |
| 2 | ANONYMOUS HUMAN | 179 | Sports |
| 3 | APOLLO TEEN | 153 | Drama |
| 4 | ARTIST COLDBLOODED | 170 | Sports |
| 5 | BEAUTY GREASE | 175 | Drama |

Query Query History

```
inner join category ct
on fc.category_id = ct.category_id
where name LIKE '%Drama%' OR name LIKE '%Sports%'

-- Question: In which category is the longest film and how long is it?
select title, length, name
from
film fm inner join film_category fc
on fm.film_id = fc.film_id
inner join category ct
on fc.category_id = ct.category_id
where name LIKE '%Drama%' OR name LIKE '%Sports%'
```

Data Output Messages Notifications

| | title text | length smallint | name text |
|---|---|---|---|
| 1 | SMOOCHY CONTROL | 184 | Sports |
| 2 | RECORDS ZORRO | 182 | Sports |
| 3 | STAR OPERATION | 181 | Sports |
| 4 | JACKET FRISCO | 181 | Drama |
| 5 | SOMETHING DUCK | 180 | Drama |

Total rows: 136 of 136 Query complete 00:00:00.149

# QUESTION 4

Query  Query History

```sql
/* 4. Create an overview of how many movies (titles) there are in each category (name) */

select count(title), name
from
film fm inner join film_category fc
on fm.film_id = fc.film_id
inner join category ct
on fc.category_id = ct.category_id
group by name

-- Question: Which category (name) is the most common among the films?
select count(title), name
from
film fm inner join film_category fc
on fm.film_id = fc.film_id
inner join category ct
on fc.category_id = ct.category_id
group by name
ORDER BY count(title)desc
-- Answer: Sports with 74 titles
-- Here, 'JOIN & GROUP BY' Query was used
```

ANSWER 4

Query  Query History

```sql
/* 4. Create an overview of how many movies (titles) there are in each category (name) *

select count(title), name
from
film fm inner join film_category fc
on fm.film_id = fc.film_id
inner join category ct
on fc.category_id = ct.category_id
group by name

-- Question: Which category (name) is the most common among the films?
```

Data Output  Messages  Notifications

| | count bigint | name text |
|---|---|---|
| 1 | 69 | Family |
| 2 | 61 | Games |
| 3 | 66 | Animation |
| 4 | 57 | Classics |
| 5 | 68 | Documentary |

Total rows: 16 of 16  Query complete 00:00:00.295

Query Query History

```sql
-- Question: Which category (name) is the most common among the films?
select name, count(title)
from
film fm inner join film_category fc
on fm.film_id = fc.film_id
inner join category ct
on fc.category_id = ct.category_id
group by name
ORDER BY count(title)desc
-- Answer: Sports with 74 titles
-- Here, 'JOIN & GROUP BY' Query was used
```

Data Output Messages Notifications

| | name text | count bigint |
|---|---|---|
| 1 | Sports | 74 |
| 2 | Foreign | 73 |
| 3 | Family | 69 |
| 4 | Documentary | 68 |
| 5 | Animation | 66 |

## QUESTION 5

Query Query History

```sql
/* 5. Create an overview of the actors' first and last names and in how many
movies they appear in   */

select * from film_actor
select * from actor
select* from film

select act.first_name, act.last_name, count(*)
from actor act inner join film_actor fa
on act.actor_id = fa.actor_id
inner join film fl
on fa.film_id = fl.film_id
group by act.first_name, act.last_name

--  Question: Which actor is part of most movies?
select act.first_name, act.last_name, count(*)
from actor act inner join film_actor fa
on act.actor_id = fa.actor_id
inner join film fl
on fa.film_id = fl.film_id
group by act.first_name, act.last_name
ORDER BY count(*) desc
```

ANSWER 5

Query Query History

```
select* from film

select act.first_name, act.last_name, count(*)
from actor act inner join film_actor fa
on act.actor_id = fa.actor_id
inner join film fl
on fa.film_id = fl.film_id
group by act.first_name, act.last_name

--  Question: Which actor is part of most movies?
select count(*), act.first_name, act.last_name
from actor act inner join film_actor fa
```

Data Output Messages Notifications

| | first_name text | last_name text | count bigint |
|---|---|---|---|
| 1 | HARRISON | BALE | 28 |
| 2 | JULIA | ZELLWEGER | 16 |
| 3 | AUDREY | OLIVIER | 25 |
| 4 | FAY | WOOD | 22 |
| 5 | GREGORY | GOODING | 30 |

Total rows: 199 of 199 Query complete 00:00:00.223

Query Query History

```
group by act.first_name, act.last_name

--  Question: Which actor is part of most movies?
select act.first_name, act.last_name, count(*)
from actor act inner join film_actor fa
on act.actor_id = fa.actor_id
inner join film fl
on fa.film_id = fl.film_id
group by act.first_name, act.last_name
ORDER BY count(*) desc
```

Data Output Messages Notifications

| | first_name text | last_name text | count bigint |
|---|---|---|---|
| 1 | SUSAN | DAVIS | 54 |
| 2 | GINA | DEGENERES | 42 |
| 3 | WALTER | TORN | 41 |
| 4 | MARY | KEITEL | 40 |
| 5 | MATTHEW | CARREY | 39 |

Total rows: 199 of 199 Query complete 00:00:00.249

# QUESTION 6

Query  Query History

```
/* 6. Create an overview of the addresses that are not associated to any customer */

select * from customer
select * from address

select * from
customer right join address
on customer.address_id = address.address_id
where customer.address_id is null

-- Question: How many addresses are that?
select COUNT(*) from
customer right join address
on customer.address_id = address.address_id
where customer.address_id is null
-- Answer: 4
-- Here, 'LEFT JOIN & FILTERING' Query was used
```

ANSWER 6

Query  Query History

```
select * from customer
select * from address

select * from
customer right join address
on customer.address_id = address.address_id
where customer.address_id is null

-- Question: How many addresses are that?
select COUNT(*) from
customer right join address
```

Data Output  Messages  Notifications

| | customer_id integer | store_id smallint | first_name text | last_name text | email text | address_id smallint | activebool boolean | create_date date | last_up timesta |
|---|---|---|---|---|---|---|---|---|---|
| 1 | [null] | [null] | [null] | [null] | [null] | [null] | [null] | [null] | [null] |
| 2 | [null] | [null] | [null] | [null] | [null] | [null] | [null] | [null] | [null] |
| 3 | [null] | [null] | [null] | [null] | [null] | [null] | [null] | [null] | [null] |
| 4 | [null] | [null] | [null] | [null] | [null] | [null] | [null] | [null] | [null] |

Total rows: 4 of 4  Query complete 00:00:00.099

Query Query History

```sql
select * from
customer right join address
on customer.address_id = address.address_id
where customer.address_id is null

-- Question: How many addresses are that?
select COUNT(*) from
customer right join address
on customer.address_id = address.address_id
where customer.address_id is null
```

Data Output Messages Notifications

| | count bigint |
|---|---|
| 1 | 4 |

## QUESTION 7

Query Query History

```sql

/* 7. Create an overview of the cities and how much sales (sum of amount)
      have occurred there  */
      -- selecting tables needed below and specifying their PRIMARY KEYS

select * from city     -- PK city id , FK country id
select * from payment  -- PK payment id, FK customer id
select * from customer -- PK customer id, FK address id
select * from address  -- PK address id, FK city id

select city, sum(amount)
from city cy inner join address ad
on cy.city_id = ad.city_id
inner join customer cs
on ad.address_id = cs.address_id
inner join payment pt
on cs.customer_id = pt.customer_id
group by city
```

ANSWER 7

Query Query History

```sql
select city, sum(amount)
from city cy inner join address ad
on cy.city_id = ad.city_id
inner join customer cs
on ad.address_id = cs.address_id
inner join payment pt
on cs.customer_id = pt.customer_id
group by city

-- Question: Which city has the most sales?
select city, sum(amount)
```

Data Output Messages Notifications

| | city text | sum numeric |
|---|---|---|
| 1 | Southport | 83.79 |
| 2 | Taguig | 99.70 |
| 3 | Tokat | 141.66 |
| 4 | Atlixco | 141.67 |
| 5 | Mukateve | 128.70 |

Query Query History

```sql
-- Question: Which city has the most sales?
select city, sum(amount)
from city cy inner join address ad
on cy.city_id = ad.city_id
inner join customer cs
on ad.address_id = cs.address_id
inner join payment pt
on cs.customer_id = pt.customer_id
group by city
ORDER BY sum(amount) desc
```

Data Output Messages Notifications

| | city text | sum numeric |
|---|---|---|
| 1 | Cape Coral | 221.55 |
| 2 | Saint-Denis | 216.54 |
| 3 | Aurora | 198.50 |
| 4 | Molodetno | 195.58 |
| 5 | Apeldoorn | 194.61 |

Total rows: 597 of 597 Query complete 00:00:00.315

# QUESTION 8

Query  Query History

```sql

/* 8. Create an overview of the revenue (sum of amount)
      grouped by a column in the format "country, city"  */
      -- selecting tables needed below and specifying their PRIMARY KEYS
select * from city     -- PK city id , FK country id
select * from payment  -- PK payment id, FK customer id
select * from customer -- PK customer id, FK address id
select * from address  -- PK address id, FK city id
select * from country  -- PK country id

select country ||',' || ' ' || city as country_city, sum(amount)
from city cy inner join address ad
on cy.city_id = ad.city_id
inner join customer cs
on ad.address_id = cs.address_id
inner join payment pt
on cs.customer_id = pt.customer_id
inner join country co
on cy.country_id = co.country_id
group by country_city

-- Question: Which country, city has the least sales?
```

ANSWER 8

Query  Query History

```sql

select country ||',' || ' ' || city as country_city, sum(amount)
from city cy inner join address ad
on cy.city_id = ad.city_id
inner join customer cs
on ad.address_id = cs.address_id
inner join payment pt
on cs.customer_id = pt.customer_id
inner join country co
on cy.country_id = co.country_id
group by country_city

```

Data Output  Messages  Notifications

| | country_city text | sum numeric |
|---|---|---|
| 1 | Germany, Siegen | 86.81 |
| 2 | Brazil, Ibirit | 97.80 |
| 3 | Runion, Saint-Denis | 216.54 |
| 4 | Vietnam, Hanoi | 119.69 |
| 5 | India, Etawah | 109.74 |

Total rows: 597 of 597  Query complete 00:00:00.187

Query Query History

```
select country ||',' || ' ' || city as country_city, sum(amount)
from city cy inner join address ad
on cy.city_id = ad.city_id
inner join customer cs
on ad.address_id = cs.address_id
inner join payment pt
on cs.customer_id = pt.customer_id
inner join country co
on cy.country_id = co.country_id
group by country_city
ORDER BY sum asc
```

Data Output Messages Notifications

| | country_city text | sum numeric |
|---|---|---|
| 1 | United States, Tallahassee | 50.85 |
| 2 | China, Fuzhou | 50.86 |
| 3 | Poland, Bydgoszcz | 52.88 |
| 4 | Sudan, al-Qadarif | 57.81 |
| 5 | Mozambique, Tete | 58.82 |

Total rows: 597 of 597 Query complete 00:00:00.155

## QUESTION 9

```
/* 9. Create a list with the average of the sales amount each staff_id has per customer *

select staff_id, customer_id,
sum(amount)as total_sum
from payment
group by staff_id, customer_id

-- next, get the average of sales amount in a sub query each staff id has per customer
-- Grouping is only by staff_id alone
SELECT staff_id, avg(total_sum)
FROM
(select staff_id, customer_id,
sum(amount)as total_sum
from payment
group by staff_id, customer_id) AS Def
GROUP BY staff_id

-- Question: Which staff_id makes on average more revenue per customer?
```

ANSWER 9

Query Query History

```sql
-- next, get the average of sales amount in a sub query each staff id has per customer
-- Grouping is only by staff_id alone
SELECT staff_id, avg(total_sum)
FROM
(select staff_id, customer_id,
sum(amount)as total_sum
from payment
group by staff_id, customer_id) AS Def
GROUP BY staff_id

-- Question: Which staff_id makes on average more revenue per customer?
SELECT staff_id, avg(total_sum)
```

Data Output Messages Notifications

| | staff_id smallint | avg numeric |
|---|---|---|
| 1 | 2 | 56.6394657762938230 |
| 2 | 1 | 55.9089649415692821 |

Total rows: 2 of 2 Query complete 00:00:00.111

Query Query History

```sql
GROUP BY staff_id

-- Question: Which staff_id makes on average more revenue per customer?
SELECT staff_id, avg(total_sum)
FROM
(select staff_id, customer_id,
sum(amount)as total_sum
from payment
group by staff_id, customer_id) AS ag
GROUP BY staff_id
ORDER BY avg desc

-- NOTE * subquery in FROM must have an alias which is "ag" above *
```

Data Output Messages Notifications

| | staff_id smallint | avg numeric |
|---|---|---|
| 1 | 2 | 56.6394657762938230 |
| 2 | 1 | 55.9089649415692821 |

# QUESTION 10

Query  Query History

```
/* 10. Create a query that shows average daily revenue of all Sundays  */

select * from payment
--First, arrive at the sum of amount by each payment date and group by payment date
select DATE(payment_date), sum(amount)
from payment
group by DATE(payment_date)

-- Let's extract week day which is Sunday as requested using the EXTRACT function
SELECT DATE(payment_date), extract(dow from payment_date), sum(amount)
FROM payment
WHERE extract(dow from payment_date) = 0
group by DATE(payment_date), extract(dow from payment_date)
-- Note 1: The group by clause on multiple columns/conditions
-- Note 2: In date documentation sunday = 0

-- To further aggregate to get the average daily revenue of sundays, a sub query will be
select ROUND(avg(total), 2)
from
(SELECT DATE(payment_date), extract(dow from payment_date), sum(amount) as total
FROM payment
WHERE extract(dow from payment_date) = 0
group by DATE(payment_date), extract(dow from payment_date)) AS yeah
```

ANSWER 10

Query  Query History

```
from payment
group by DATE(payment_date)

-- Let's extract week day which is Sunday as requested using the EXTRACT function
SELECT DATE(payment_date), extract(dow from payment_date), sum(amount)
FROM payment
WHERE extract(dow from payment_date) = 0
group by DATE(payment_date), extract(dow from payment_date)
-- Note 1: The group by clause on multiple columns/conditions
-- Note 2: In date documentation sunday = 0

-- To further aggregate to get the average daily revenue of sundays, a sub query will be
select ROUND(avg(total), 2)
```

Data Output  Messages  Notifications

| | date<br>date | extract<br>numeric | sum<br>numeric |
|---|---|---|---|
| 1 | 2020-01-26 | 0 | 736.30 |
| 2 | 2020-02-16 | 0 | 1346.77 |
| 3 | 2020-03-01 | 0 | 2724.49 |
| 4 | 2020-03-22 | 0 | 2602.72 |
| 5 | 2020-04-05 | 0 | 273.36 |

Total rows: 7 of 7  Query complete 00:00:00.092

Query Query History

```sql
group by DATE(payment_date), extract(dow from payment_date)
-- Note 1: The group by clause on multiple columns/conditions
-- Note 2: In date documentation sunday = 0

-- To further aggregate to get the average daily revenue of sundays, a sub query will be
select ROUND(avg(total), 2)
from
(SELECT DATE(payment_date), extract(dow from payment_date), sum(amount) as total
FROM payment
WHERE extract(dow from payment_date) = 0
group by DATE(payment_date), extract(dow from payment_date)) AS yeah
/*    Note sum(amount) should have an alias so as the sub query also,
```

Data Output Messages Notifications

| | round numeric |
|---|---|
| 1 | 1423.05 |

Total rows: 1 of 1 Query complete 00:00:00.112

## QUESTION 11

Query Query History

```sql

/* 11. Create a list of movies - with their length and their replacement cost
       that are longer than the average length in each replacement cost group  */

select title, length, replacement_cost
from film

-- This is a 'correlated sub query question'

select fl1.title, fl1.length, replacement_cost
from film fl1
where length >
(select avg(length)
from film fl2
where fl1.replacement_cost = fl2.replacement_cost)
group by replacement_cost, fl1.title,  fl1.length

-- Question: Which two movies are the shortest on that list and how long are they?
```

ANSWER 11

Query  Query History

```
-- This is a 'correlated sub query question'

select fl1.title, fl1.length, replacement_cost
from film fl1
where length >
(select avg(length)
from film fl2
where fl1.replacement_cost = fl2.replacement_cost)
group by replacement_cost, fl1.title,  fl1.length

-- Question: Which two movies are the shortest on that list and how long are they?
```

Data Output  Messages  Notifications

| | title text | length smallint | replacement_cost numeric (5,2) |
|---|---|---|---|
| 1 | CONTROL ANTHEM | 185 | 9.99 |
| 2 | DUDE BLINDNESS | 132 | 9.99 |
| 3 | EDGE KISSING | 153 | 9.99 |
| 4 | ENCINO ELF | 143 | 9.99 |
| 5 | FACTORY DRAGON | 144 | 9.99 |

Total rows: 490 of 490  Query complete 00:00:00.757

Query  File  Query History

```
group by replacement_cost, fl1.title,  fl1.length

-- Question: Which two movies are the shortest on that list and how long are they?
select title, length, replacement_cost
from film fl1
where length >
(select avg(length)
from film fl2
where fl1.replacement_cost = fl2.replacement_cost)
ORDER BY length asc
```

Data Output  Messages  Notifications

| | title text | length smallint | replacement_cost numeric (5,2) |
|---|---|---|---|
| 1 | CELEBRITY HORN | 110 | 24.99 |
| 2 | SEATTLE EXPECATIONS | 110 | 18.99 |
| 3 | WONDERLAND CHRISTMAS | 111 | 19.99 |
| 4 | GROOVE FICTION | 111 | 13.99 |
| 5 | WIND PHANTOM | 111 | 12.99 |

Total rows: 490 of 490  Query complete 00:00:00.990

## QUESTION 12

Query  Query History

```
/*  12. Create a list that shows the "average customer lifetime value"
        grouped by the different districts  */

select customer.customer_id, district, sum(amount) as total
from
customer inner join payment
on customer.customer_id = payment.customer_id
inner join address
on
customer.address_id = address.address_id
group by customer.customer_id, district
order by district
```

ANSWER 12

Query  Query History

```
SELECT district, ROUND(AVG(total), 2)
from
(select customer.customer_id, district, sum(amount) as total
from
customer inner join payment
on customer.customer_id = payment.customer_id
inner join address
on
customer.address_id = address.address_id
group by customer.customer_id, district
order by district) AS dub
GROUP BY district
```

Data Output  Messages  Notifications

| | district text | round numeric |
|---|---|---|
| 1 | | 136.00 |
| 2 | Abu Dhabi | 107.22 |
| 3 | Aceh | 134.66 |
| 4 | Adana | 89.79 |
| 5 | Addis Abeba | 91.77 |

Total rows: 376 of 376  Query complete 00:00:00.213

Query Query History

```sql
SELECT district, ROUND(AVG(total), 2) as avg_value
from
(select customer.customer_id, district, sum(amount) as total
from
customer inner join payment
on customer.customer_id = payment.customer_id
inner join address
on
customer.address_id = address.address_id
group by customer.customer_id, district
order by district) AS dub
GROUP BY district
ORDER BY avg_value desc
```

Data Output Messages Notifications

| | district text | avg_value numeric |
|---|---|---|
| 1 | Saint-Denis | 216.54 |
| 2 | Minsk | 195.58 |
| 3 | Skikda | 173.63 |
| 4 | Khartum | 169.65 |

Total rows: 376 of 376 Query complete 00:00:00.313

## QUESTION 13

```sql

/*  13. Create a list that shows all payments including the payment_id, amount, and
    the film category (name) plus the total amount that was made in this category.
    Order the results ascendingly by the category (name)
    and as second order criterion by the payment_id ascendingly  */

select payment_id,
amount,
title,
name
from payment

-- needed attributes below
-- Join sequence and primary keys
select* from payment       -- PK payment_id, FK rental_id (SELECT attributes needed here
select * from rental       -- PK rental_id, FK inventory_id (bridge/join connection table
select * from inventory    -- PK inventory_id, FK film_id (bridge/join connection table)
select * from film         -- PK film_id  (SELECT attributes needed here is iii. TITLE)
select* from film_category -- PK film_id, FK category_id (bridge/join connection table)
select * from category     -- PK category_id (SELECT attributes needed here is iv. NAME)

-- First, a left join to return all from payment table join to the rental table
```

ANSWER 13

Query Query History

```
select name, p.payment_id, sum(amount) as total_amount
from payment p left join rental r
on p.rental_id = r.rental_id
left join inventory iy
on r.inventory_id = iy.inventory_id
left join film f
on iy.film_id = f.film_id
left join film_category fc
on f.film_id = fc.film_id
left join category ct
on fc.category_id = ct.category_id
GROUP BY name, p.payment_id
ORDER BY name asc, p.payment_id asc
```

Data Output Messages Notifications

| | name text | payment_id integer | total_amount numeric |
|---|---|---|---|
| 1 | Action | 16055 | 2.99 |
| 2 | Action | 16073 | 10.99 |
| 3 | Action | 16075 | 4.99 |
| 4 | Action | 16093 | 4.99 |

Total rows: 1000 of 16049 Query complete 00:00:00.373

Query Query History

```
select name, sum(amount) as total_amount
from payment p left join rental r
on p.rental_id = r.rental_id
left join inventory iy
on r.inventory_id = iy.inventory_id
left join film f
on iy.film_id = f.film_id
left join film_category fc
on f.film_id = fc.film_id
left join category ct
on fc.category_id = ct.category_id
GROUP BY name
ORDER BY name asc
```

Data Output Messages Notifications

| | name text | total_amount numeric |
|---|---|---|
| 1 | Action | 4375.85 |
| 2 | Animation | 4656.30 |
| 3 | Children | 3655.55 |
| 4 | Classics | 3639.59 |

Total rows: 16 of 16 Query complete 00:00:00.105

## QUESTION 14

Query  Query History

```
/* 14  Create a list with the top overall revenue of a film title (sum of amount per
       title) for each category (name)  */

select name, f. title, sum(amount) as total_amount
from payment p left join rental r
on p.rental_id = r.rental_id
left join inventory iy
on r.inventory_id = iy.inventory_id
left join film f
on iy.film_id = f.film_id
left join film_category fc
on f.film_id = fc.film_id
left join category ct
on fc.category_id = ct.category_id
GROUP BY name, f.title

-- Question: Which is the top-performing film in the animation category?
```

## ANSWER 14

Query  Query History

```
select name, f. title, sum(amount) as total_amount
from payment p left join rental r
on p.rental_id = r.rental_id
left join inventory iy
on r.inventory_id = iy.inventory_id
left join film f
on iy.film_id = f.film_id
left join film_category fc
on f.film_id = fc.film_id
left join category ct
on fc.category_id = ct.category_id
GROUP BY name, f.title
```

Data Output  Messages  Notifications

| | name text | title text | total_amount numeric |
|---|---|---|---|
| 1 | Drama | TREATMENT JEKYLL | 12.93 |
| 2 | Music | SCALAWAG DUCK | 172.68 |
| 3 | Classics | LOVER TRUMAN | 62.90 |
| 4 | Animation | CLUB GRAFFITI | 44.80 |

Total rows: 958 of 958  Query complete 00:00:00.409

Data Output Messages Notifications

| | name text | title text | total_amount numeric |
|---|---|---|---|
| 1 | Music | TELEGRAPH VOYAGE | 231.73 |
| 2 | Documentary | WIFE TURN | 223.69 |
| 3 | Comedy | ZORRO ARK | 214.69 |
| 4 | Sci-Fi | GOODFELLAS SALUTE | 209.69 |
| 5 | Sports | SATURDAY LAMBS | 204.72 |
| 6 | Drama | TORQUE BOUND | 198.72 |
| 7 | Foreign | INNOCENT USUAL | 191.74 |
| 8 | Travel | BUCKET BROTHERHOOD | 180.66 |
| 9 | Family | RANGE MOONWALKER | 179.73 |
| 10 | Games | MASSACRE USUAL | 179.70 |
| 11 | Animation | DOGMA FAMILY | 178.70 |
| 12 | Action | FOOL MOCKINGBIRD | 175.77 |
| 13 | New | MAIDEN HOME | 163.76 |
| 14 | Horror | LOLA AGENT | 159.76 |
| 15 | Children | BACKLASH UNDEFEATED | 158.81 |
| 16 | Classics | STEEL SANTA | 141.77 |

Total rows: 16 of 16 Query complete 00:00:05.236 Ln 518, Col 1

I ultimately completed this project here. I covered some handier functions in SQL, but I will make others in the next SQL posts. Thank you and I hope that this project could be helpful to my fellow analyst.