131. NOTE: JOIN combines columns and UNION combines rows.

132. NOTE: UNION combines multiple SELECT statements to give final table output.

133. NOTE: UNION:

- Columns are not matched by their names.
- Columns are matched by the order they appear in SELECT statement.
- Column names of first table will be used in result/final table.
- Data types of the column to be matched MUST be SAME.
- Number of columns of the SELECT statements taking part in UNION must be SAME.
- We can UNION multiple SELECT statements.
- Exact same rows in multiple/same tables/table in UNION will appear only once in resulting table.
- No duplicates in UNION.
- Use 'UNION ALL' instead of 'UNION' if you want duplicates to appear in final table.

132. Fetch all firstName of actors and customers in single table.

```
SELECT first_name,'actor' as origin
FROM actor
UNION ALL
SELECT first_name, 'customer'
FROM customer
ORDER BY first_name
```

133. NOTE: A subquery is used when you want result of one query 'the subquery' to be evaluated first and passed on to parent/main query to get final result.

134. Fetch all the payment that has amount strictly greater than the average payment amount.

```
SELECT *
FROM payment
WHERE amount > (SELECT AVG(amount) FROM payment)
```

135. Fetch all payments made by customer with firstName 'Adam'.

```
SELECT *
FROM payment
WHERE customer_id = (
    SELECT customer_id
    FROM customer
    WHERE first_name = 'ADAM'
)
```

136. NOTE: Use 'IN' operator to match multiple values if subquery has multiple output values.

137. Fetch payment record of all customers whose firstName starts with 'A'.

```
SELECT *
FROM payment
WHERE customer_id IN (
      SELECT customer_id
    FROM customer
    WHERE first_name LIKE 'A%'
)
ORDER BY customer_id
```

138. Fetch all films where the length is longer than average of all films.

```
SELECT film_id,title,length
FROM film
WHERE length > (SELECT AVG(length) FROM film)
ORDER BY length
```

139. Fetch all film titles which are available in store 2 (inventory) more than 3 times.

```
SELECT film_id,title
FROM film
WHERE film_id IN (
      SELECT film_id
      FROM inventory
      WHERE store_id=2
      GROUP BY film_id
      HAVING COUNT(*)>3
)
ORDER BY film_id
```

140. Fetch customer details of all customers who made a payment on '2020-01-25'.

```
SELECT customer_id,first_name,last_name
FROM customer
WHERE customer_id IN (
      SELECT customer_id
      FROM payment
      WHERE DATE(payment_date) = '2020-01-25'
)
```

141. Fetch customer details of all customers who have spent more than $30 total.

```
SELECT customer_id,first_name,last_name,email
FROM customer
WHERE customer_id IN (
      SELECT customer_id
      FROM payment
      GROUP BY customer_id
      HAVING SUM(amount)>30
)
ORDER BY customer_id
```

142. Fetch details of customers who are from district 'California' and have made

payment of total more than $100.

```
SELECT customer_id,first_name,last_name
FROM customer
WHERE address_id IN (
      SELECT address_id
      FROM address
      WHERE district='California'
) AND customer_id IN (
      SELECT customer_id
      FROM payment
      GROUP BY customer_id
      HAVING SUM(amount)>100
)
ORDER BY first_name,last_name
```

143. NOTE: We can also use SELECT FROM statement on a subquery.

144. NOTE: Subquery must have a alias name to be used with SELECT FROM

statement.

145. Fetch the average of total lifetime spend of individual customers from payment table.

```
SELECT ROUND(AVG(total_amount),2) AS avg_lifetime_spend
FROM (
      SELECT SUM(amount) AS total_amount
      FROM payment
      GROUP BY customer_id
) AS sum_table
```

146. Fetch the average of daily spend of customers from payment table.

```
SELECT ROUND(AVG(total_amount),2) AS avg_daily_spend
FROM (
      SELECT SUM(amount) AS total_amount
      FROM payment
      GROUP BY DATE(payment_date)
) AS sum_table
```

147. NOTE: We can us subquery right after SELECT keyword if subquery returns 1 value.

148. Fetch the difference between each payment and the maximum payment from the payment table.

```
SELECT *,
      (SELECT MAX(amount) FROM payment)-amount as difference
FROM payment
```

149. NOTE: Correlated subquery, as the name hints the subquery's result is evaluated as per the row values of the main/parent query.

150. NOTE: Correlated subquery runs for each row of main/parent query.

151. Fetch the payments where the amount is highest for each individual customer.

```
SELECT *
FROM payment as p1
WHERE p1.amount = (
      SELECT MAX(amount)
      FROM payment as p2
      WHERE p2.customer_id=p1.customer_id
      GROUP BY p2.customer_id
)
ORDER BY p1.customer_id
```

152. Fetch the fil details of film that has lowest replacement cost among the films having same rating category.

```
SELECT film_id,title,replacement_cost,rating
FROM film as f1
WHERE f1.replacement_cost = (
      SELECT MIN(replacement_cost)
      FROM film as f2
      WHERE f1.rating = f2.rating
      GROUP BY f2.rating
)
ORDER BY rating
```

152. Fetch the fil details of film that has longest length among the films having same rating category.

```
SELECT film_id,title,length,rating
FROM film as f1
WHERE f1.length = (
      SELECT MAX(length)
      FROM film as f2
      WHERE f1.rating = f2.rating
      GROUP BY f2.rating
)
ORDER BY rating
```

153. Fetch all payments and in front of each payment show the maximum amount spent by that customer.

```
SELECT *,
(
      SELECT MAX(amount)
      FROM payment as p2
      WHERE p1.customer_id=p2.customer_id
      GROUP BY p2.customer_id
) as max_spent
FROM payment as p1
ORDER BY p1.customer_id
```

154. Fetch all payments and in front of each payment show the total amount spent and total payments made by that customer.

```
SELECT *,
(
      SELECT SUM(amount)
      FROM payment as p2
      WHERE p1.customer_id=p2.customer_id
) as total_spent,
(
      SELECT COUNT(amount)
      FROM payment as p2
      WHERE p1.customer_id=p2.customer_id
) as payment_count
FROM payment as p1
ORDER BY p1.customer_id, p1.amount DESC
```

154. Fetch only those film details where their replacement cost is highest in their rating category and the average replacement cost of that rating category.

```
SELECT f1.film_id,f1.title,f1.replacement_cost,f1.rating,
(
      SELECT ROUND(AVG(replacement_cost),2)
      FROM film as f3
      WHERE f3.rating=f1.rating
) as avg_replacement_cost
FROM film as f1
WHERE f1.replacement_cost = (
      SELECT MAX(replacement_cost)
      FROM film as f2
      WHERE f1.rating = f2.rating
)
```

155. Fetch only payment details where the amount is maximum for the customer with
same first name.

```
SELECT p1.payment_id,
      p1.customer_id,
      c1.first_name,
      c1.last_name,
      p1.amount
FROM payment as p1
INNER JOIN customer as c1
ON p1.customer_id=c1.customer_id
WHERE p1.amount = (
      SELECT MAX(p2.amount)
      FROM payment as p2
      INNER JOIN customer as c2
      ON p2.customer_id = c2.customer_id
      WHERE c1.first_name = c2.first_name
)
ORDER BY c1.first_name
```