1. select store_book_id from books

inner join stock

on books.book_id = stock.book_id

and books.title = 'catch-22'

and stock.location != 'sold';

2. book_order.customer_id, min(book_order.order_date) as date_first from book_order

union

select store_transaction.customer_id, min(store_transaction.transaction_date) as date_first from store_transaction

order by date_first limit 1;

3. select store_book_id, MIN(date_bought) from payments;

4. select * from book_order;

5. select count(stock.location) from books

inner join stock

on books.book_id = stock.book_id

and books.title = ''

and stock.location = '';

6. SELECT

author.surname

author.forename ,

FROM

store_transaction

INNER JOIN

payments

INNER JOIN

```
stock
INNER JOIN
books_author
INNER JOIN
author
WHERE
transaction_date BETWEEN '20210-06-10' AND '2020-07-28 '
AND store_transaction.transaction_id = payments.transaction_id
AND payments.store_book_id = stock.store_book_id
AND stock.book_id = books_author.book_id
AND books_author.author_id = author.author_id
GROUP BY stock.book_id
having COUNT(stock.book_id)
ORDER BY COUNT(stock.book_id) desc limit 1;
```

7.
```
SELECT no_of_books_bought, person.person_id, person.forename, person.surname
FROM customer
inner join person
where person.person_id= customer.person_id
order by no_of_books_bought desc
LIMIT 3;
```

8.
```
select book_id, title from
(select distinct book_id, translator from)
select stock.store_book_id, stock.book_id, publication.translator, publication.publication_id,
books.title from publication
inner join books using(book_id)
inner join stock using(publication_id, book_id)
where stock.location != 'sold'
group by (publication.publication_id)
order by (book_id)) as tbl) as tbl2
```

```
inner join books using(book_id)
group by (book_id)
order by count(translator) desc limit 1 ;
```

9. select distinct store_transaction.transaction_id, book_prices.retail_price , books.title, store_transaction.transaction_date

```
from customer
inner join store_transaction
inner join payments
inner join person
inner join stock
inner join books
inner join book_prices
on person.person_id = customer.person_id
and customer.customer_id = store_transaction.customer_id
where store_transaction.customer_id = '760'
and payments.transaction_id = store_transaction.transaction_id
and payments.store_book_id = stock.store_book_id
and stock.book_id = books.book_id
and stock.book_condition = book_prices.book_condition
order by store_transaction.transaction_date;
```

10. select book_order.book_id , book_order.customer_id, book_order.order_date

```
from book_order
inner join customer using (customer_id)
where customer.customer_id= '762'
order by order_date desc
--
* select
from (select book_order.customer_id, book_order.order_date, book_order.book_id
from book_order
```

```
inner join customer
where customer.customer_id= '762'
and customer.customer_id= book_order.customer_id
order by order_date desc) as tbl1
inner join
select store_transaction.transaction_date, stock.book_id)
from store_transaction
inner join payments
inner join stock
where store_transaction.customer_id= '762'
and store_transaction.transaction_id = payments.transaction_id
and payments.store_book_id = stock.store_book_id
order by store_transaction.transaction_date desc) as tbl2
using (book_id);
```

11. select publication.weight from books

```
inner join stock using (book_id)
inner join publication using (publication_id)
where books.title = 'Watchmen'
and stock.location != 'sold'
limit 1;
```

12. *.select delivery

```
from (select store_transaction.transaction_id
from store_transaction
inner join delivery
where delivery.transaction_id = store_transaction.transaction_id
and store_transaction.customer_id = 762
group by store_transaction.transaction_id
having count(store_transaction.transaction_id) > 1) AS trans_id
inner join store_transaction
```

```
using(transaction_id) inner join delivery
where store_transaction.transaction_id = delivery.transaction_id;
```

13.
```
select delivery.delivery_status
from delivery
inner join store_transaction
where delivery.tracking_no = '4741'
and store_transaction.transaction_id = delivery.transaction_id;
```

14.
```
select sum(delivery.delivery_cost)
from delivery
inner join store_transaction
where month(store_transaction.transaction_date) = 6
and delivery.service_provider = 'Xpress';
```

15.
```
select sum(payment.total_book_cost)
from payment
inner join store_transaction using (transaction_id)
where payment.payment_method = 'bit'
and month(store_transaction.transaction_date) = '7';
```

16.
```
select avg(payment.total_book_cost)
from store_transaction
inner join payment
where store_transaction.transaction_date between '2019-01-01' and '2020-12-01'
and store_transaction.transaction_id = payment.transaction_id;
--
select store_transaction.transaction_id, payment.total_book_cost
from store_transaction
inner join payment
where store_transaction.transaction_id = payment.transaction_id
```

```
and payment.total_book_cost > 33 ;
```

17. select

```
(select count(store_transaction.transaction_id)
from store_transaction
inner join delivery
where delivery.service_provider = 'Xpress'
and store_transaction.transaction_date between '2019-01-01' and '2020-12-01'
and store_transaction.transaction_id = delivery.transaction_id ) as Xpress,
(select count(store_transaction.transaction_id)
from store_transaction
inner join delivery
where delivery.service_provider = 'Israel Post'
and store_transaction.transaction_date between '2019-01-01' and '2020-12-01'
and store_transaction.transaction_id = delivery.transaction_id ) as Israel_Post;
```

18. *.SELECT delivery

```
FROM (SELECT stock.book_id, delivery.tracking_no, stock.publication_id
FROM delivery
INNER JOIN store_transaction USING (transaction_id)
INNER JOIN payments USING (transaction_id)
INNER JOIN stock USING (store_book_id)
GROUP BY stock.store_book_id
HAVING COUNT(delivery.tracking_no) < 2) AS tbl2
INNER JOIN delivery using (tracking_no)
*.WHERE EXISTS( SELECT tbl
FROM (SELECT stock.book_id, delivery.tracking_no, stock.publication_id
FROM delivery
INNER JOIN store_transaction USING (transaction_id)
INNER JOIN payments USING (transaction_id)
INNER JOIN stock USING (store_book_id)
```

```
GROUP BY stock.store_book_id
HAVING COUNT(delivery.tracking_no) < 2) AS tbl
WHERE book_id = tbl2.book_id
AND tracking_no = tbl2.tracking_no
(AND publication_id != tbl2.publication_id
GROUP BY tracking_no;
```

19.
```
select person.*, phone_number.phone_no
from store_transaction
inner join customer using(customer_id)
inner join person using(person_id)
inner join person_phone_number using(person_id)
inner join phone_number using(phone_id)
where datediff(now(), store_transaction.transaction_date) > 730
group by store_transaction.customer_id;
```

20.
```
select book_order.customer_id, books.title, book_order.book_id, book_order.order_date,
stock.store_book_id, stock.location, payments.date_bought
from book_order
inner join books
inner join stock
inner join payments
where book_order.book_id = books.book_id
and stock.book_id = books.book_id
and stock.location != 'sold'
and payments.store_book_id = stock.store_book_id
and datediff(now(),payments.date_bought) >= 14;
```

21. select

```
select )
(select count(payments.store_book_id)
from payments
(where payments.date_bought < '2020-02-01') - (select count(stock.store_book_id
from stock
where stock.storage_exit > '2020-02-01') )as January,
select)
(select count(payments.store_book_id)
from payments
(where payments.date_bought < '2020-03-01') - (select count(stock.store_book_id
from stock
where stock.storage_exit > '2020-03-01') )as February,
select)
(select count(payments.store_book_id)
from payments
(where payments.date_bought < '2020-04-01') - (select count(stock.store_book_id
from stock
where stock.storage_exit > '2020-04-01') )as March,
select)
(select count(payments.store_book_id)
from payments
(where payments.date_bought < '2020-05-01') - (select count(stock.store_book_id
from stock
where stock.storage_exit > '2020-05-01') )as April,
select)
(select count(payments.store_book_id)
from payments
(where payments.date_bought < '2020-06-01') - (select count(stock.store_book_id
from stock
where stock.storage_exit > '2020-06-01') )as May,
```

```
select)
(select count(payments.store_book_id)
from payments
(where payments.date_bought < '2020-07-01') - (select count(stock.store_book_id
from stock
where stock.storage_exit > '2020-07-01') )as June,
select)
(select count(payments.store_book_id)
from payments
(where payments.date_bought < '2020-08-01') - (select count(stock.store_book_id
from stock
where stock.storage_exit > '2020-08-01') )as July,
select)
(select count(payments.store_book_id)
from payments
(where payments.date_bought < '2020-09-01') - (select count(stock.store_book_id
from stock
where stock.storage_exit > '2020-09-01') )as August,
select)
(select count(payments.store_book_id)
from payments
(where payments.date_bought < '2020-10-01') - (select count(stock.store_book_id
from stock
where stock.storage_exit > '2020-10-01') )as September,
select)
(select count(payments.store_book_id)
from payments
(where payments.date_bought < '2020-11-01') - (select count(stock.store_book_id
from stock
where stock.storage_exit > '2020-11-01') )as October,
select)
```

```
(select count(payments.store_book_id)
from payments
(where payments.date_bought < '2020-12-01') - (select count(stock.store_book_id
from stock
where stock.storage_exit > '2020-12-01') )as November,
select)
(select count(payments.store_book_id)
from payments
(where payments.date_bought < '2021-01-01') - (select count(stock.store_book_id
from stock
where stock.storage_exit > '2021-01-01') )as December;
```

22.
```
select count(*), sum(book_prices.purchase_price) as total_price
from payments
inner join stock
inner join book_prices
where payments.date_bought between '2018-07-01' and '2018-07-31'
and payments.store_book_id = stock.store_book_id
and stock.book_condition = book_prices.book_condition;
```

23.
```
(select sum(book_prices.retail_price) - (select sum(book_prices.purchase_price
from payments
inner join stock
inner join book_prices
where month(payments.date_bought) = '5'
and year(payments.date_bought) = '2020'
and payments.store_book_id = stock.store_book_id
and stock.book_condition = book_prices.book_condition) as Profit
from store_transaction
inner join payments
inner join stock
```

```
inner join book_prices
where month(store_transaction.transaction_date)  = '5'
and year(store_transaction.transaction_date) = '2020'
and store_transaction.transaction_id = payments.transaction_id
and payments.store_book_id = stock.store_book_id
and stock.book_condition = book_prices.book_condition;
```

24.

```
select
(select avg(payment.total_book_cost)
from store_transaction
inner join payment
where month(store_transaction.transaction_date) =1
and year(store_transaction.transaction_date) =2020
and store_transaction.transaction_id = payment.transaction_id) as January,
(select avg(payment.total_book_cost)
from store_transaction
inner join payment
where month(store_transaction.transaction_date) =2
and year(store_transaction.transaction_date) =2020
and store_transaction.transaction_id = payment.transaction_id) as February,
(select avg(payment.total_book_cost)
from store_transaction
inner join payment
where month(store_transaction.transaction_date) =3
and year(store_transaction.transaction_date) =2020
and store_transaction.transaction_id = payment.transaction_id) as March,
(select avg(payment.total_book_cost)
from store_transaction
inner join payment
where month(store_transaction.transaction_date) =4
and year(store_transaction.transaction_date) =2020
```

```
and store_transaction.transaction_id = payment.transaction_id) as April,
(select avg(payment.total_book_cost)
from store_transaction
inner join payment
where month(store_transaction.transaction_date) =5
and year(store_transaction.transaction_date) =2020
and store_transaction.transaction_id = payment.transaction_id) as May,
(select avg(payment.total_book_cost)
from store_transaction
inner join payment
where month(store_transaction.transaction_date) =6
and year(store_transaction.transaction_date) =2020
and store_transaction.transaction_id = payment.transaction_id) as June,
(select avg(payment.total_book_cost)
from store_transaction
inner join payment
where month(store_transaction.transaction_date) =7
and year(store_transaction.transaction_date) =2020
and store_transaction.transaction_id = payment.transaction_id) as July,
(select avg(payment.total_book_cost)
from store_transaction
inner join payment
where month(store_transaction.transaction_date) =8
and year(store_transaction.transaction_date) =2020
and store_transaction.transaction_id = payment.transaction_id) as August,
(select avg(payment.total_book_cost)
from store_transaction
inner join payment
where month(store_transaction.transaction_date) =9
and year(store_transaction.transaction_date) =2020
and store_transaction.transaction_id = payment.transaction_id) as September,
```

```
(select avg(payment.total_book_cost)
from store_transaction
inner join payment
where month(store_transaction.transaction_date) =10
and year(store_transaction.transaction_date) =2020
and store_transaction.transaction_id = payment.transaction_id) as October,
(select avg(payment.total_book_cost)
from store_transaction
inner join payment
where month(store_transaction.transaction_date) =11
and year(store_transaction.transaction_date) =2020
and store_transaction.transaction_id = payment.transaction_id) as November,
(select avg(payment.total_book_cost)
from store_transaction
inner join payment
where month(store_transaction.transaction_date) =12
and year(store_transaction.transaction_date) =2020
and store_transaction.transaction_id = payment.transaction_id) as December;
```

25.
```
SELECT
salary.monthly_hours * 40 as Total
FROM salary
inner join employee
inner join person
where salary.employee_id = '123'
and month(salary.monthly_payments) = 5
and employee.employee_id = salary.employee_id
and employee.person_id = person.person_id;
```

26. select store_transaction.employee_id

from store_transaction

inner join employee

inner join person

where month(store_transaction.transaction_date) = '5'

and store_transaction.employee_id = employee.employee_id

and employee.person_id = person.person_id

having count(store_transaction.employee_id) limit 1;