

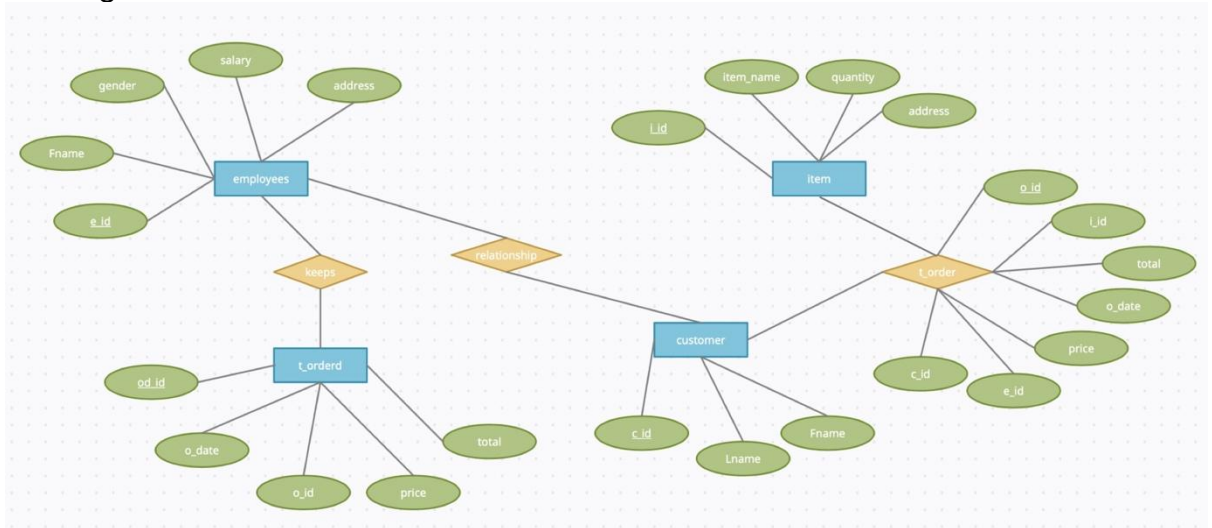
Part 1

1. Suggest a situation where you can use a database to manage and record daily transactions.

A restaurant consists of number of customers, employees and orders. Each customer offers several orders. A number of order details make up each order. Each order contains order ID, order date, the items price and the total price. The employees in the restaurant is an important part of preparing the item of the restaurant, the employees make customer orders according to order details.

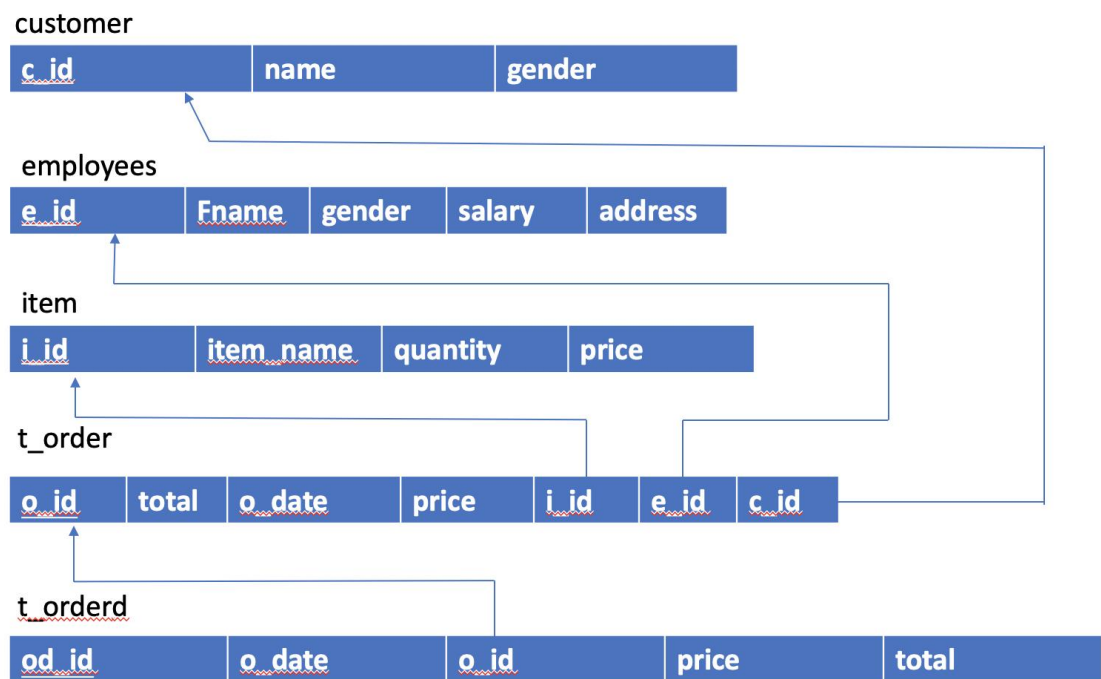
However, the employees may need to make changes to item details based on customer needs. This can help the restaurant to get a higher ratings and increase the exposure of the restaurant on the delivery app, which can lead to more orders. Each order detail is very similar to the constituent elements of order, but is different. Orders can be generated on a weekly, monthly or yearly basis for each order detail. At the same time, the order details can only show the order details of each customer, if the restaurant need to count the monthly orders or annual orders, we need the orders as the total data support. The data for each order detail can help calculate the most and least sold items for each month, as well as the difference in sales for different time periods.

2. ER diagram



3. Conceptual design into a relational model

Hansung_Restaurant



Part 2

1. Create the corresponding database using DDL and
2. Create all the necessary tables identified above using DDL

res* Administration - Data Import/Restore

Limit to 1000 rows

```
40 foreign key(i_id)
41 references item(i_id),
42 foreign key(e_id)
43 references employees(e_id),
44 foreign key(c_id)
45 references customer(c_id)
46 );
47
48 • create table t_orderd(
49     od_id int,
50     o_date datetime,
51     o_id int,
52     price decimal(4,2),
53     total decimal(4,2),
54     primary key(od_id),
55     foreign key(o_id)
56     references t_order(o_id)
57 );
```

100% 3:57

Action Output

	Time	Action	Response
✓ 5	19:35:50	create table customer(...	0 row(s) affected
⚠ 6	19:35:50	create table employees...	0 row(s) affected, 1 warning(s): 1681 Specifying number of digits for floating point data types is de
✓ 7	19:35:50	create table item(i_id i...	0 row(s) affected
✓ 8	19:35:50	create table t_order(o...	0 row(s) affected
✓ 9	19:35:50	create table t_orderd(...	0 row(s) affected

3. Populate at least three of your tables with some data using DML

The screenshot displays a database management tool interface. At the top, there are tabs for different tables: **res**, **customer***, **employees***, **item**, **t_order***, and **t_orderd**. Below the tabs is a toolbar with various icons and a "Limit to 1000 rows" dropdown. The main area contains a SQL script with the following lines:

```
60 values (51,'le','ke');
61 • select*from customer;
62
63 • insert into employees(e_id,Fname)
64 values(20,'shi');
65 • select*from employees;
66
67 #insert into item(i_id,item_name)
68 #values (60,'Chicken');
69 • insert into item (i_id, item_name, quantity, price) values (1, 'Chicken Mayo', 109, 10.59);
70 • insert into item (i_id, item_name, quantity, price) values (2, 'Kimchi Dishes', 13, 11.4);
71 • insert into item (i_id, item_name, quantity, price) values (3, 'Fried Chiecken', 562, 15.35);
```

Below the script, the status bar shows "100%", "3:54", and "1 error found". The "Result Grid" section displays a table with columns **c_id**, **Lname**, and **Fname**. The first row shows values 51, le, and ke. Below this, there are three rows with NULL values. The "Action Output" section shows a log of actions and their responses:

	Time	Action	Response
✓ 40	14:18:23	select from employees LIMIT 0, 1000	1 row(s) returned
✓ 41	14:18:23	insert into item (i_id, item_name, quantity, pr...	1 row(s) affected
✓ 42	14:18:23	insert into item (i_id, item_name, quantity, pr...	1 row(s) affected
✓ 43	14:18:23	insert into item (i_id, item_name, quantity, pr...	1 row(s) affected
✓ 44	14:18:23	select*from item LIMIT 0, 1000	3 row(s) returned

4.
1).

```
7      #Q2
8  • ⊖ create table customer(
9      c_id int primary key,
10     Lname varchar(60),
11     Fname varchar(60)
12 );
13
14  • ⊖ create table employees(
15     e_id int,
16     Fname varchar(20),
17     gender varchar(60),
18     salary float(5,2),
19     address varchar(60),
20     primary key(e_id)
21 );
22
23  • ⊖ create table item(
24     i_id int,
```


100% 16:11


Action Output


	Time	Action	Response
✓ 85	19:57:27	insert into customer (c_id, Lname, Fname) values (57, 'Echlin', 'Gilbert...	1 row(s) affected
✓ 86	19:57:27	insert into customer (c_id, Lname, Fname) values (58, 'Kivlin', 'Lila')	1 row(s) affected
✓ 87	19:57:27	insert into customer (c_id, Lname, Fname) values (59, 'Wakeley', 'Hals...	1 row(s) affected
✓ 88	19:57:27	insert into customer (c_id, Lname, Fname) values (60, 'Ibbitson', 'Myrt...	1 row(s) affected


2).

```
60 • insert into employees (e_id, Fname, gender, salary, address)
61 • select*from employees;
```


100%  122:60

Result Grid 

 Filter Rows:

Export: 

	e_id	Fname	gender	salary	address	
▶	1	Pamela	Bigender	4327.5100000000	15 Melrose Circle	
	2	Rickie	Male	3894.3300000000	116 Columbus Street	
	3	Mellisent	Female	4967.9800000000	92 Brown Terrace	
	4	Asia	Female	4112.6500000000	50 Harper Junction	
	5	Barret	Male	5108.5100000000	68887 Sheridan Circle	
	6	Atlante	Female	6495.5200000000	184 Lotheville Pass	
	7	Obadias	Polygender	4890.1900000000	39493 Eagle Crest Place	
	8	Sampson	Bigender	6845.1100000000	95 Chive Road	
employees 1						

Action Output 

		Time	Action	Response
✓	249	21:09:53	insert into employees (e_id, Fname, gender, salary, a...	1 row(s) affected
✓	250	21:09:53	insert into employees (e_id, Fname, gender, salary, a...	1 row(s) affected
✓	251	21:09:53	insert into employees (e_id, Fname, gender, salary, a...	1 row(s) affected
✓	252	21:09:53	select*from employees LIMIT 0, 1000	60 row(s) returned

3).

```
60 • insert into item (i_id, item_name, quantity, price) values (  
61 • select*from item;
```

100% 18:61

Result Grid

Filter Rows: Search

Export:

i_id	item_name	quantity	price	
3	BB Tinted	562	15.350000	
4	Preparing	121	17.280000	
5	DawnMist	557	17.870000	
6	Pollens	293	18.320000	
7	FCHICKEN DUCK	479	9.580000	
8	Inspra	363	19.820000	
9	Perrigo Hydroquinone	587	6.110000	
10	hydrochlorothiazide	32	7.100000	

item 1

Action Output

	Time	Action	Response
✓ 334	21:18:43	insert into item (i_id, item_name, quantity, price) valu...	1 row(s) affected
✓ 335	21:18:43	insert into item (i_id, item_name, quantity, price) valu...	1 row(s) affected
✓ 336	21:18:43	insert into item (i_id, item_name, quantity, price) valu...	1 row(s) affected
✓ 337	21:18:43	select*from item LIMIT 0, 1000	60 row(s) returned

4).

```

29
30 • ○ create table t_order(
31     o_id int primary key,
32     total decimal(60,6),
33     o_date varchar(60),
34     price decimal(60,6),
35     i_id int,
36     e_id int,
37     c_id int,
38     foreign key(i_id)
39     references item(i_id),
40     foreign key(e_id)
41     references employees(e_id),
42     foreign key(c_id)
43     references customer(c_id)
44 );
45

```

100%	1:1		
Action Output			
	Time	Action	Response
✓ 281	22:44:57	insert into t_order (o_id, total, o_date, price, i_id, e_id, c_id) values (58,...	1 row(s) affected
✓ 282	22:44:57	insert into t_order (o_id, total, o_date, price, i_id, e_id, c_id) values (59,...	1 row(s) affected
✓ 283	22:44:57	insert into t_order (o_id, total, o_date, price, i_id, e_id, c_id) values (60,...	1 row(s) affected
✓ 284	22:44:57	select*from t_order LIMIT 0, 1000	60 row(s) returned

5).

```

45
46 • ○ create table t_orderd(
47     od_id int primary key,
48     o_date varchar(60),
49     o_id int,
50     price decimal(10,2),
51     total decimal(10,2),
52     foreign key(o_id)
53     references t_order(o_id)
54 );

```

100%	1:1		
Action Output			
	Time	Action	Response
✓ 421	22:46:54	insert into t_orderd (od_id, o_date, o_id, price, total) values (57, '10/09/...	1 row(s) affected
✓ 422	22:46:54	insert into t_orderd (od_id, o_date, o_id, price, total) values (58, '09/02/...	1 row(s) affected
✓ 423	22:46:54	insert into t_orderd (od_id, o_date, o_id, price, total) values (59, '11/20/...	1 row(s) affected
✓ 424	22:46:54	insert into t_orderd (od_id, o_date, o_id, price, total) values (60, '04/04/...	1 row(s) affected

Part 3

1. Show the total number of transactions your database is storing and, depending on your database, the most sold/listed item or customer with the highest number of purchases.

The screenshot shows a database management tool interface with a top toolbar containing icons for file operations, execution, and search. Below the toolbar is a tabbed interface with tabs for 'res*', 'customer*', 'employees*', 'item', 't_order*', and 't_orderd'. The main area displays SQL queries with line numbers 73 through 84. The queries are as follows:

```
73
74 #Part 3
75 #Q1.1
76 #Show the total number of transactions your database is storing
77 • select sum(quantity) as 'Best Selling Dishes' from item;
78
79 #Q1.2
80 #Show the most sold/listed item or customer with the highest number of p
81 • select quantity from item
82 where quantity = (select max(quantity) from item);
83
84 #Q2
85 select i_id, price from item
```

Below the queries is a 'Result Grid' section with a search bar and an 'Export' button. The grid shows the results of the first query, with a single row containing the value '109' under the 'quantity' column. Below the grid is an 'Action Output' section with a table showing the execution history:

	Time	Action	Response
✓ 22	23:20:44	insert into item (i_id, item_name, quantity, price) values (1, 'Chicken Ma...	1 row(s) affected
✓ 23	23:20:53	select*from employees LIMIT 0, 1000	0 row(s) returned
✓ 24	23:23:04	select*from t_order LIMIT 0, 1000	0 row(s) returned
✓ 25	23:23:18	select quantity from item where quantity = (select max(quantity) from i...	1 row(s) returned

2. Write a query statement that includes “Order by” and “Group by”.

The screenshot shows a database management tool interface with a top toolbar containing icons for file operations, execution, and search. Below the toolbar is a tabbed interface with tabs for 'res*', 'customer*', 'employees*', 'item', 't_order*', and 't_orderd'. The 'item' tab is active, displaying a SQL editor with the following queries:

```
79 #Q1.2
80 #Show the most sold/listed item or customer with the highest number of
81 • select quantity from item
82 where quantity = (select max(quantity) from item);
83
84 #Q2
85 • select i_id, price from item
86 group by i_id order by price DESC;
87
88 #Q3
89 • select i_id, count(i_id) from item
90 where i_id = 'price';
```

Below the editor is a 'Result Grid' section with a search bar and an 'Export' button. It displays a table with two columns: 'i_id' and 'price'. The first row shows '1' and '10.590000'. Below the table is a section labeled 'item 56'.

At the bottom is an 'Action Output' section with a table showing the execution of four queries:

	Time	Action	Response
✓ 23	23:20:53	select*from employees LIMIT 0, 1000	0 row(s) returned
✓ 24	23:23:04	select*from t_order LIMIT 0, 1000	0 row(s) returned
✓ 25	23:23:18	select quantity from item where quantity = (select max(quantity) from i...	1 row(s) returned
✓ 26	23:24:10	select i_id, price from item group by i_id order by price DESC LIMIT 0, 1...	1 row(s) returned

3. Write a query statement that uses pattern matching

The screenshot shows a database IDE with a top toolbar containing icons for file operations, execution, and search. Below the toolbar is a tab bar with tabs for 'res*', 'customer*', 'employees*', 'item', 't_order*', and 't_orderd'. The main editor area contains SQL code with line numbers 80 through 92. The code includes a subquery for finding the maximum quantity, followed by two queries labeled #Q2 and #Q3. Query #Q2 selects item ID and price, grouped by item ID and ordered by price in descending order. Query #Q3 selects item ID and count of item ID, filtered by item ID = 'price'. Query #Q4 is partially visible and appears to be an inner join. Below the editor is a 'Result Grid' section with a search bar and an 'Export' button. The grid shows a single row with 'i_id' 60 and 'price' NULL. Below the grid is a section labeled 'item 28'. At the bottom is an 'Action Output' table showing the execution history of the queries.

```
80 where quantity = (select max(quantity) from item);
81
82 #Q2
83 • select i_id, price from item
84 group by i_id order by price DESC;
85
86 #Q3
87 • select i_id, count(i_id) from item
88 where i_id = 'price';
89
90 #Q4 ??????
91 • select od_id,o_date,total from t_orderd
92 inner join t_order on t_orderd.od_id=t_order.o_id
```

100% 1:89

Result Grid Filter Rows: Search Export:

i_id	price
60	NULL

item 28

Action Output

	Time	Action	Response
✓	207 22:31:09	select*from t_order LIMIT 0, 1000	0 row(s) returned
✓	208 22:34:40	select*from t_order LIMIT 0, 1000	0 row(s) returned
✓	209 22:35:24	select quantity from item where quantity = (select m...	0 row(s) returned
✓	210 22:35:33	select i_id, price from item group by i_id order by pri...	1 row(s) returned

4. Show information from three tables based on criteria of your choice

The screenshot shows a database IDE interface with a top toolbar containing icons for file operations, execution, and search. Below the toolbar is a tabbed interface with tabs for 'res', 'customer*', 'employees*', 'item', 't_order*', and 't_orderd'. The main editor area displays three SQL queries:

```
97      #Q3
98      •  select i_id, count(i_id) from item
99      where i_id = 'price';
100
101      #Q4
102      •  select od_id from t_orderd
103      inner join t_order on t_orderd.od_id=t_order.o_id
104      inner join item on t_orderd.o_id=item.item_name;
105
106      #Q5
107      •  select o_id, count(o_id)
108      as 'Most Popular Dishes' from t_order
```

Below the editor, a status bar shows '100%' zoom, a refresh icon, '49:104' characters, and '1 error found'. The 'Result Grid' section is visible, showing a table with the following data:

od_id
2

5. Create a view that includes information from the most frequent seven transactions

The screenshot shows a database management tool interface with a top toolbar containing icons for file operations, execution, and search. Below the toolbar is a SQL editor with the following queries:

```
96
97 #Q4
98 • select od_id from t_orderd
99   inner join t_order on t_orderd.od_id=t_order.o_id
100   inner join item on t_orderd.o_id=item.item_name;
101
102 #Q5
103 • select o_id, count(o_id)
104   as 'Most Popular Dishes' from t_order
105   group by o_id
106   order by count(o_id) desc limit 7;
107
```

Below the editor is a 'Result Grid' section showing the results of the queries. The first query (Q4) has one result row:

o_id	Most Popular Dishes
1	1

The second query (Q5) has 64 results, with the first row highlighted:

o_id	Most Popular Dishes
1	1

Below the result grid is an 'Action Output' section showing a log of database actions:

	Time	Action	Response
✓ 59	23:44:22	insert into t_order (o_id, total, o_date, price, i_id, e_id, c_id) values (1, 8...	1 row(s) affected
✓ 60	23:47:19	insert into t_orderd (od_id, o_date, o_id, price, total) values (1, '02/16/2...	1 row(s) affected
✓ 61	23:47:30	select od_id from t_orderd inner join t_order on t_orderd.od_id=t_order....	0 row(s) returned
✓ 62	23:52:40	select o_id, count(o_id) as 'Most Popular Dishes' from t_order group by...	1 row(s) returned

6. 1). Shows the total number of transactions with corresponding details every month

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following queries:

```
110 group by o_id
111 order by count(o_id) desc limit 7;
112
113 #Q6.1
114 • select month(t_orderd.o_date)
115 as 'month', sum(t_orderd.total),
116 group_concat(od_id)
117 as 'Monthly Transactions' from t_orderd
118 group by month
119 order by month;
120
121 #Q6.2
```

The result grid shows the output of the query:

month	sum(t_orderd.total)	Monthly Transactions
6	65939.70000	65939.700000

- 2). Shows customer purchase value per month

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following queries:

```
119 order by month;
120
121 #Q6.2
122 • select month(t_orderd.o_date)
123 as 'month', sum(item.price*item.quantity)
124 as 'Purchase Value' from item inner join t_orderd
125 on t_orderd.od_id=item.i_id
126 group by month
127 order by month;
128
129
130 #Q6.3
```

The result grid shows the output of the query:

month	Purchase Value
1	20672.700000
6	37201.654000

The Action Output panel shows the following actions:

Time	Action	Response
20:12:25	insert into item (i_id, item_name, quantity, pr...	1 row(s) affected
20:12:25	insert into item (i_id, item_name, quantity, pr...	1 row(s) affected
20:12:25	insert into item (i_id, item_name, quantity, pr...	1 row(s) affected
20:12:25	select*from item LIMIT 0. 1000	3 row(s) returned

3). Shows name of product and number sold each month

The screenshot shows a database query editor with a toolbar at the top containing icons for file operations, execution, and search. Below the toolbar, a SQL query is entered in a text area. The query is as follows:

```
127     order by month;
128
129
130     #Q6.3
131     • select month(t_order.o_date)
132       as 'month',
133       item.item_name, item.i_id, item.quantity from item
134     inner join t_order
135     on item.i_id = t_order.o_id
136     order by month;
137
138
```

Below the query editor, there is a status bar showing '100%' zoom and '16:136' dimensions. Below that is a 'Result Grid' section with a 'Filter Rows' search bar and an 'Export' button. The result grid displays the following data:

month	item_name	i_id	quantity
3	Chicken Mayo	1	60