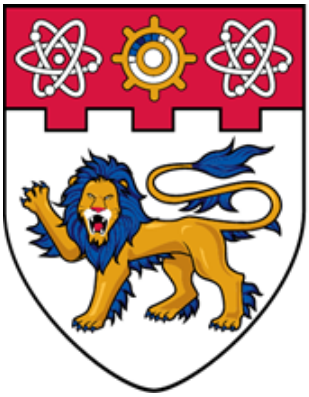

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

20S1 CZ2007 Project: Lab 3 Submission

SS9 Group 1

U1922940J - Le Quang Anh (Leader)

U1922557L - Chew Harris Rezal

U1922222E - Max Pang Liang Hui

U1922050A - Neo Yong Tai

U1922981C - Ong Jing Hong Elliott

U1923230B - Suhana Gupta

Section 1: SQL DDL Queries

To aid us in making the queries, we used Excel to generate some fake data. This allowed us to create a lot of rows and also help us check whether our queries return the expected results.

Shops

```
CREATE TABLE shops (
	shop_id		INT		NOT NULL	IDENTITY(1,1),
	shop_name		VARCHAR(255)	NOT NULL	UNIQUE,

	CONSTRAINT PK_shops PRIMARY KEY (shop_id)
);

INSERT INTO shops (shop_name)
VALUES
	('World of Sport'),
	('Challenger'),
	('Asos'),
	('Zalora'),
	('M1'),
	('Starhub'),
	('Royal Sporting'),
	('Courts'),
	('Harvey Norman');
```

Makers

```
CREATE TABLE makers (
	maker_id		INT		NOT NULL	IDENTITY(1,1),
	maker_name		VARCHAR(255)	NOT NULL	UNIQUE,

	CONSTRAINT PK_makers PRIMARY KEY (maker_id)
);

INSERT INTO makers (maker_name)
VALUES
	('Apple'),
	('Samsung'),
	('Sony'),
```

```
	('Philips'),
	('King Koil'),
	('Simmons'),
	('Uniqlo'),
	('H&M'),
	('Nike'),
	('Adidas');
```

Categories

```
CREATE TABLE categories (
	category_id		INT		NOT NULL	IDENTITY(1,1),
	category_name		VARCHAR(255)	NOT NULL	UNIQUE,

	CONSTRAINT PK_categories PRIMARY KEY (category_id)
);

INSERT INTO categories (category_name)
VALUES
	('Mobile Phones'),
	('Furniture'),
	('Home Appliances'),
	('Apparels'),
	('Sports Wear'),
	('Sports Equipment'),
	('Bags');
```

Products

```
CREATE TABLE products (
	product_id		INT		NOT NULL	IDENTITY(1,1),
	product_name		VARCHAR(255)	NOT NULL,
	maker_id		INT		NOT NULL,
	category_id		INT		NOT NULL,

	CONSTRAINT PK_products PRIMARY KEY (product_id),

	CONSTRAINT FK_products_maker_id	FOREIGN KEY (maker_id)	REFERENCES
makers(maker_id),
```

```
	CONSTRAINT FK_products_category_id FOREIGN KEY (category_id) REFERENCES
categories(category_id)
);

INSERT INTO products (product_name, maker_id, category_id) -- 1 to 35
VALUES
	('iPhone 12', 1, 1),
	('iPhone 11', 1, 1),
	('iPhone X', 1, 1),
	('iPhone 8', 1, 1),
	('Galaxy S10', 2, 1),
	('Galaxy S20', 2, 1),
	('Spring Mattress', 5, 2),
	('Spinal Guard Mattress', 6, 2),
	('Samsung 55 Inches TV', 2, 3),
	('Samsung 22 Inches TV', 2, 3),
	...;
```

**In the actual table, we have 35 rows*

Products in Shops

```
CREATE TABLE products_in_shops (
	shop_id		INT		NOT NULL,
	product_id	INT		NOT NULL,
	stock		INT		NOT NULL,
	price		DECIMAL(10,2)	NOT NULL,

	CONSTRAINT PK_products_in_shops PRIMARY KEY (product_id, shop_id),

	CONSTRAINT FK_products_in_shops_product_id FOREIGN KEY (product_id) REFERENCES
products(product_id),
	CONSTRAINT FK_products_in_shops_shop_id	FOREIGN KEY (shop_id)	REFERENCES
shops(shop_id)
);

INSERT INTO products_in_shops (shop_id, product_id, stock, price)
VALUES
	(1, 22, 4, 300),
```

```
	(1, 23, 141, 55),
	(1, 24, 68, 150),
	(1, 25, 100, 78),
	(1, 26, 65, 47),
	(1, 27, 0, 58),
	(1, 28, 27, 71),
	(1, 29, 0, 49),
	(1, 30, 108, 15),
	(2, 09, 93, 680),
	...;
```

**In the actual table, we have 92 rows*

Records

```
CREATE TABLE records (
	shop_id		INT		NOT NULL,
	product_id		INT		NOT NULL,
	start_date		DATETIME	NOT NULL,
	end_date		DATETIME	NULL,
	price		DECIMAL(10,2)	NOT NULL,

	CONSTRAINT PK_records PRIMARY KEY (product_id, shop_id, start_date),

	CONSTRAINT FK_records_product_id FOREIGN KEY (product_id) REFERENCES
products(product_id),
	CONSTRAINT FK_records_shop_id	FOREIGN KEY (shop_id)	REFERENCES
shops(shop_id)
);

INSERT INTO records (shop_id, product_id, start_date, end_date, price)
VALUES
	(1, 22, '2020-04-10 00:00:00', NULL, 300),
	(1, 23, '2020-04-10 00:00:00', NULL, 55),
	(1, 24, '2020-04-10 00:00:00', NULL, 150),
	(1, 25, '2020-04-10 00:00:00', NULL, 78),
	(1, 26, '2020-04-10 00:00:00', NULL, 47),
	(1, 27, '2020-04-10 00:00:00', NULL, 58),
	(1, 28, '2020-04-10 00:00:00', NULL, 71),
```

```
	(1, 29, '2020-04-10 00:00:00', NULL, 49),
	(1, 30, '2020-04-10 00:00:00', NULL, 15),
	(2, 09, '2020-04-10 00:00:00', NULL, 680),
	...;
```

In the actual table, we have 92 rows

Statuses

```
CREATE TABLE statuses (
	status_id		INT		NOT NULL	IDENTITY(1,1),
	status_name		VARCHAR(255)	NOT NULL	UNIQUE,

	CONSTRAINT PK_statuses PRIMARY KEY (status_id),
);

INSERT INTO statuses (status_name)
VALUES
	('Being Processed'),
	('Shipped'),
	('Delivered'),
	('Returned'),
	('Pending'),
	('Being Handled'),
	('Addressed');
```

Users

```
CREATE TABLE users (
	user_id		INT		NOT NULL	IDENTITY(1,1),
	name		VARCHAR(255)	NOT NULL,

	CONSTRAINT PK_users PRIMARY KEY (user_id)
);

INSERT INTO users (name)
VALUES
	('Bart'),
	('Billy'),
	('Bobby'),
```

```
        ('Danny'),
        ('Izzy'),
        ('Jey'),
        ('Jimmy'),
        ('Matty'),
        ('Phillip'),
        ('Timmy'),
        ...;
```

**In the actual table, we have 1,042 rows*

Orders

```
CREATE TABLE orders (
        order_id              INT             NOT NULL   IDENTITY(1,1),
        user_id               INT             NOT NULL,
        ordered_date          DATETIME        NOT NULL,
        shipping_address      VARCHAR(1000)   NOT NULL,
        total_cost            DECIMAL(10,2)   NOT NULL,

        CONSTRAINT PK_orders PRIMARY KEY (order_id),

        CONSTRAINT FK_orders_user_id FOREIGN KEY (user_id) REFERENCES users(user_id)
);

INSERT INTO orders (user_id, ordered_date, shipping_address, total_cost)
VALUES
        (369, '2020-04-01 09:34:08', 'Address of user ID 369', 230),
        (408, '2020-04-01 09:49:49', 'Address of user ID 408', 1160),
        (460, '2020-04-01 10:03:48', 'Address of user ID 460', 150),
        (350, '2020-04-01 11:10:11', 'Address of user ID 350', 57),
        (405, '2020-04-01 12:02:36', 'Address of user ID 405', 51),
        (642, '2020-04-01 12:13:32', 'Address of user ID 642', 1800),
        (216, '2020-04-01 16:00:12', 'Address of user ID 216', 190),
        (323, '2020-04-01 17:02:59', 'Address of user ID 323', 2160),
        (139, '2020-04-01 17:26:01', 'Address of user ID 139', 499),
        (886, '2020-04-01 18:43:21', 'Address of user ID 886', 255),
        ...;
```

In the actual table, we have 5,358 rows

Products in Orders

```
CREATE TABLE products_in_order (
	order_id		INT		NOT NULL,
	shop_id			INT		NOT NULL,
	product_id		INT		NOT NULL,
	quantity		INT		NOT NULL,
	status_id		INT		NOT NULL,
	delivery_date		DATETIME	NULL,
	price			DECIMAL(10,2)	NOT NULL,

	CONSTRAINT PK_products_in_order PRIMARY KEY (product_id, order_id, shop_id),

	CONSTRAINT FK_products_in_order_product_id FOREIGN KEY (product_id) REFERENCES
products(product_id),
	CONSTRAINT FK_products_in_order_order_id   FOREIGN KEY (order_id)   REFERENCES
orders(order_id),
	CONSTRAINT FK_products_in_order_shop_id    FOREIGN KEY (shop_id)    REFERENCES
shops(shop_id),
	CONSTRAINT FK_products_in_order_status_id  FOREIGN KEY (status_id)  REFERENCES
statuses(status_id)
);

INSERT INTO products_in_order (order_id, shop_id, product_id, quantity, status_id,
delivery_date, price)
VALUES
	(1, 8, 07, 1, 3, '2020-05-22 17:48:12', 230),
	(2, 3, 35, 2, 3, '2020-05-24 09:56:27', 580),
	(3, 1, 24, 1, 3, '2020-05-25 12:40:19', 150),
	(4, 7, 30, 3, 3, '2020-05-25 10:31:18', 19),
	(5, 3, 21, 1, 3, '2020-05-24 10:13:26', 51),
	(6, 9, 13, 3, 3, '2020-05-22 14:13:38', 600),
	(7, 7, 24, 1, 3, '2020-05-22 10:13:35', 190),
	(8, 9, 02, 2, 3, '2020-05-23 16:48:43', 1080),
	(9, 1, 29, 1, 3, '2020-05-24 09:03:45', 49),
	(9, 1, 24, 3, 3, '2020-05-23 15:12:32', 150),
	...;
```

In the actual table, we have 21,961 rows

Rates

```
CREATE TABLE rates (
	user_id		INT		NOT NULL,
	product_id		INT		NOT NULL,
	rating		INT		NOT NULL,
	date_time		DATETIME	NOT NULL,
	comment		VARCHAR(1000)	NULL,

	CONSTRAINT PK_rates PRIMARY KEY (user_id, product_id),

	CONSTRAINT FK_rates_user_id	FOREIGN KEY (user_id)	REFERENCES
users(user_id),
	CONSTRAINT FK_rates_product_id FOREIGN KEY (product_id) REFERENCES
products(product_id)
);

INSERT INTO rates (user_id, product_id, rating, date_time, comment)
VALUES
	(01, 04, 2, '2020-09-09 06:10:46', ''),
	(01, 05, 1, '2020-10-24 00:48:21', ''),
	(01, 06, 1, '2020-08-23 17:30:39', ''),
	(01, 07, 5, '2020-07-22 04:21:57', 'Great'),
	(01, 08, 3, '2020-10-17 01:34:21', ''),
	(01, 09, 5, '2020-07-24 15:33:24', ''),
	(01, 11, 1, '2020-07-18 16:18:23', ''),
	(01, 12, 1, '2020-10-30 00:39:59', ''),
	(01, 13, 3, '2020-07-05 07:14:51', 'Okay'),
	(01, 14, 3, '2020-07-27 07:01:54', ''),
	...;
```

In the actual table, we have 17,112 rows

Employees

```
CREATE TABLE employees (
```

```
	employee_id		INT		NOT NULL	IDENTITY(1,1),
	name			VARCHAR(255)	NOT NULL,
	monthly_salary		DECIMAL(10,2)	NOT NULL,

	CONSTRAINT PK_employees PRIMARY KEY (employee_id)
);

INSERT INTO employees (name, monthly_salary)
VALUES
	('Sammy', 2300),
	('Shea', 2200),
	('Scott', 3000),
	('Steve', 2300),
	('Sally', 2900),
	('Selene', 2400),
	('Shayna', 2300),
	('Sheena', 2200),
	('Shirlena', 2200),
	('Susana', 2600),
	...;
```

**In the actual table, we have 58 rows*

Complaints

```
CREATE TABLE complaints (
	complaint_id		INT		NOT NULL	IDENTITY(1,1),
	user_id			INT		NOT NULL,
	employee_id		INT		NULL,
	status_id		INT		NOT NULL,
	filed_date_time		DATETIME	NOT NULL,
	addressed_date_time	DATETIME	NULL,
	complaint_text		VARCHAR(1000)	NOT NULL,

	CONSTRAINT PK_complaints PRIMARY KEY (complaint_id),

	CONSTRAINT FK_complaints_user_id	FOREIGN KEY (user_id)	REFERENCES
users(user_id),
```

```
	CONSTRAINT FK_complaints_employee_id FOREIGN KEY (employee_id) REFERENCES
employees(employee_id),
	CONSTRAINT FK_complaints_status_id   FOREIGN KEY (status_id)   REFERENCES
statuses(status_id)
);

INSERT INTO complaints (user_id, employee_id, status_id, filed_date_time,
addressed_date_time, complaint_text)
VALUES
	(894, NULL, 5, '2020-06-01 16:00:29', NULL, 'Did not receive in time'),
	(784, 6, 6, '2020-06-06 17:45:27', NULL, 'Too expensive'),
	(607, 4, 7, '2020-06-10 10:45:50', '2020-07-03 09:35:08', 'Did not receive in
time'),
	(790, 9, 6, '2020-06-13 09:27:39', NULL, 'Worse than expected'),
	(603, NULL, 5, '2020-06-20 10:12:52', NULL, 'Worse than expected'),
	(519, NULL, 5, '2020-06-21 11:57:33', NULL, 'Hate it'),
	(326, 34, 7, '2020-06-22 10:27:33', '2020-07-20 18:16:42', 'Product sucks'),
	(263, 4, 7, '2020-06-24 10:10:08', '2020-07-17 13:24:23', 'Do not purchase'),
	(1031, NULL, 5, '2020-06-24 11:30:29', NULL, 'Delivery was delayed'),
	(822, NULL, 5, '2020-06-25 10:43:41', NULL, 'Hate it'),
	...;
```

**In the actual table, we have 872 rows*

Complaints on Order

```
CREATE TABLE complaints_on_order (
	complaint_id		INT		NOT NULL,
	order_id		INT		NOT NULL,

	CONSTRAINT PK_complaints_on_order PRIMARY KEY (complaint_id),

	CONSTRAINT FK_complaints_on_order_complaint_id   FOREIGN KEY (complaint_id)
REFERENCES complaints(complaint_id),
	CONSTRAINT FK_complaints_on_order_order_id	FOREIGN KEY (order_id)
REFERENCES orders(order_id)
);

INSERT INTO complaints_on_order (complaint_id, order_id)
```

```
VALUES
	(4, 199),
	(5, 147),
	(6, 121),
	(7, 81),
	(8, 67),
	(10, 205),
	(13, 344),
	(15, 556),
	(16, 659),
	(18, 848),
	...;
```

In the actual table, we have 558 rows

Complaints on Shop

```
CREATE TABLE complaints_on_shop (
	complaint_id		INT		NOT NULL,
	shop_id			INT		NOT NULL,

	CONSTRAINT PK_complaints_on_shop PRIMARY KEY (complaint_id),

	CONSTRAINT FK_complaints_on_shop_complaint_id	FOREIGN KEY (complaint_id)
REFERENCES complaints(complaint_id),
	CONSTRAINT FK_complaints_on_shop_shop_id	FOREIGN KEY (shop_id)
REFERENCES shops(shop_id)
);

INSERT INTO complaints_on_shop (complaint_id, shop_id)
VALUES
	(1, 9),
	(2, 9),
	(3, 8),
	(9, 7),
	(11, 2),
	(12, 4),
	(14, 2),
	(17, 1),
```

```
    (19, 2),
    (20, 1),
    ...;
```

*In the actual table, we have 314 rows

Section 2: Images of the Tables

We have included screenshots of the first ten rows for each table, taken from the SQL server.

Shops

	shop_id	shop_name
2	2	Challenger
3	8	Courts
4	9	Harvey No...
5	5	M1
6	7	Royal Spo...
7	6	Starhub
8	1	World of ...
9	4	Zalora

Makers

	maker_id	maker_name
1	10	Adidas
2	1	Apple
3	8	H&M
4	5	King Koil
5	9	Nike
6	4	Philips
7	2	Samsung
8	6	Simmons

Categories

	category_id	category_name
1	4	Apparels
2	7	Bags
3	2	Furniture
4	3	Home Appliances
5	1	Mobile Phones
6	6	Sports Equipment
7	5	Sports Wear

Products

	product_id	product_name	maker_id	category_id
1	1	iPhone 12	1	1
2	2	iPhone 11	1	1
3	3	iPhone Xs	1	1
4	4	iPhone 8	1	1
5	5	Galaxy S10	2	1
6	6	Galaxy S20	2	1
7	7	Spring Mattr...	5	2
8	8	Spinal Guar...	6	2

Products in Shop

	shop_id	product_id	stock	price
1	5	1	188	1150.00
2	6	1	30	1270.00
3	8	1	147	1180.00
4	9	1	9	1240.00
5	5	2	0	920.00
6	6	2	112	920.00
7	8	2	135	880.00
8	9	2	117	1080.00

Records

	shop_id	product_id	start_date	end_date	price
4	9	1	2020-04-10 00:00:00.000	NULL	1240.00
5	5	2	2020-04-10 00:00:00.000	NULL	920.00
6	6	2	2020-04-10 00:00:00.000	NULL	920.00
7	8	2	2020-04-10 00:00:00.000	NULL	880.00
8	9	2	2020-04-10 00:00:00.000	NULL	1080.00
9	5	3	2020-04-10 00:00:00.000	NULL	560.00
10	6	3	2020-04-10 00:00:00.000	NULL	520.00
11	8	3	2020-04-10 00:00:00.000	NULL	520.00

Statuses

	status_id	status_name
1	7	Addressed
2	6	Being Handled
3	1	Being Proces...
4	3	Delivered
5	5	Pending
6	4	Returned
7	2	Shipped

Users

	user_id	name
10	10	Tim...
11	11	Bri...
12	12	Bro...
13	13	Dari...
14	14	Dar...
15	15	Darryl
16	16	De...
17	17	De...

Orders

	order_id	user_id	ordered_date	shipping_address	total_cost
1	1	369	2020-04-01 09:34:08.000	Address of user ID 369	230.00
2	2	408	2020-04-01 09:49:49.000	Address of user ID 408	1160.00
3	3	460	2020-04-01 10:03:48.000	Address of user ID 460	150.00
4	4	350	2020-04-01 11:10:11.000	Address of user ID 350	57.00
5	5	405	2020-04-01 12:02:36.000	Address of user ID 405	51.00
6	6	642	2020-04-01 12:13:32.000	Address of user ID 642	1800.00
7	7	216	2020-04-01 16:00:12.000	Address of user ID 216	190.00
8	8	323	2020-04-01 17:02:59.000	Address of user ID 323	2160.00

Products in Orders

	order_id	shop_id	product_id	quantity	status_id	delivery_date	price
1	46	9	1	1	3	2020-05-24 10:11:43.000	1240.00
2	86	5	1	3	3	2020-05-24 12:40:11.000	1150.00
3	115	8	1	3	3	2020-05-22 12:55:35.000	1180.00
4	149	6	1	2	3	2020-05-23 14:27:36.000	1270.00
5	170	6	1	3	3	2020-05-24 11:31:21.000	1270.00
6	182	5	1	2	3	2020-05-22 13:50:10.000	1150.00
7	217	9	1	3	3	2020-05-25 13:01:21.000	1240.00
8	245	8	1	2	3	2020-05-23 10:30:43.000	1180.00

Rates

	user_id	product_id	rating	date_time	comment
1	1	4	2	2020-09-09 06:10:46.000	
2	1	5	1	2020-10-24 00:48:21.000	
3	1	6	1	2020-08-23 17:30:39.000	
4	1	7	5	2020-07-22 04:21:57.000	Great
5	1	8	3	2020-10-17 01:34:21.000	
6	1	9	5	2020-07-24 15:33:24.000	
7	1	11	1	2020-07-18 16:18:23.000	
8	1	12	1	2020-10-30 00:39:59.000	

Employees

	employee_id	name	monthly_salary
1	1	Sammy	2300.00
2	2	Shea	2200.00
3	3	Scott	3000.00
4	4	Steve	2300.00
5	5	Sally	2900.00
6	6	Selene	2400.00
7	7	Shayna	2300.00
8	8	Shee...	2200.00

Complaints

	complaint_id	user_id	employee_id	status_id	filed_date_time	addressed_date_time	complaint_text
1	1	894	NULL	5	2020-06-01 16:00:29.000	NULL	Did not receive in time
2	2	784	6	6	2020-06-06 17:45:27.000	NULL	Too expensive
3	3	607	4	7	2020-06-10 10:45:50.000	2020-07-03 09:35:08.000	Did not receive in time
4	4	790	9	6	2020-06-13 09:27:39.000	NULL	Worse than expected
5	5	603	NULL	5	2020-06-20 10:12:52.000	NULL	Worse than expected
6	6	519	NULL	5	2020-06-21 11:57:33.000	NULL	Hate it
7	7	326	34	7	2020-06-22 10:27:33.000	2020-07-20 18:16:42.000	Product sucks
8	8	263	4	7	2020-06-24 10:10:08.000	2020-07-17 13:24:23.000	Do not purchase

Complaints on Orders

	complaint_id	order_id
1	4	199
2	5	147
3	6	121
4	7	81
5	8	67
6	10	205
7	13	344
8	15	556

Complaints on Shops

	complaint_id	shop_id
1	1	9
2	2	9
3	3	8
4	9	7
5	11	2
6	12	4
7	14	2
8	17	1

Section 3: Sample Queries

In this section, we present our SQL queries for the statements in Appendix B. For each query, we have included a screenshot of the query result and the Excel result.

1. Find the average price of “iPhone Xs” on Sharkee from 1 August 2020 to 31 August 2020.

a) SQL query

```
SELECT AVG(R.price) AS avg_price
FROM products P, records R
WHERE P.product_name = 'iPhone Xs'
      AND P.product_id = R.product_id
      AND (R.start_date BETWEEN '2020-08-01' AND '2020-08-31'
           OR R.end_date BETWEEN '2020-08-01' AND '2020-08-31'
           OR (R.start_date < '2020-08-01' AND (R.end_date > '2020-08-31'
                                                OR R.end_date IS NULL)));
```

b) SQL result

	avg_price
1	550.000000

c) Excel result

550

2. Find products that received at least 100 ratings of “5” in August 2020, and order them by their average ratings.

a) SQL query

```
SELECT product_id, ROUND(AVG(CAST(rating AS float)), 2) AS average_rating
FROM rates
WHERE MONTH(date_time) = 8 AND YEAR(date_time) = 2020
      AND product_id IN (SELECT product_id
                         FROM rates
                         WHERE MONTH(date_time) = 8 AND YEAR(date_time) = 2020
                               AND rating = 5
                         GROUP BY product_id
                         HAVING COUNT(rating) >= 100)
GROUP BY product_id
ORDER BY average_rating;
```

b) SQL result (note that this is in ascending order)

	product_id	average_rating
1	21	3.97
2	14	4.03
3	28	4.1
4	35	4.1
5	7	4.13

c) Excel result (note that this is in descending order)

prod id	no. ratings	>= 100	avg. rating
7	111	TRUE	4.13
35	117	TRUE	4.1
28	130	TRUE	4.1
14	117	TRUE	4.03
21	111	TRUE	3.97

3. For all products purchased in June 2020 that have been delivered, find the average time from the ordering date to the delivery date.

a) SQL query

```
SELECT AVG(DATEDIFF(DAY, O.ordered_date, PIO.delivery_date)) AS average_time
FROM orders O, products_in_order PIO
WHERE MONTH(O.ordered_date) = 6 AND YEAR(O.ordered_date) = 2020
      AND PIO.delivery_date IS NOT NULL
      AND O.order_id = PIO.order_id;
```

b) SQL result

	average_time
1	29

c) Excel result

29

4. Let us define the “latency” of an employee by the average that he/she takes to process a complaint. Find the employee with the smallest latency.

a) SQL query

```
SELECT TOP (1) WITH TIES E.employee_id, E.name
FROM complaints C, employees E
WHERE E.employee_id = C.employee_id
      AND C.status_id = 7
GROUP BY E.employee_id, E.name
ORDER BY AVG(CAST(C.addressed_date_time-C.filed_date_time AS float)) ASC;
```

b) SQL result

	employee_id	name	avg_latency
1	49	Sullivan	6.56116512345679

c) Excel result

id	name	avg days
49	Sullivan	6.5

5. Produce a list that contains (i) all products made by Samsung, and (ii) for each of them, the number of shops on Sharkee that sell the product.

a) SQL query

```
SELECT P.product_id, P.product_name, COUNT(shop_id) AS num_shop
FROM products P, products_in_shops PIS, makers M
WHERE M.maker_name = 'Samsung'
      AND P.product_id = PIS.product_id
      AND P.maker_id = M.maker_id
GROUP BY P.product_id, P.product_name;
```

b) SQL result

	product_id	product_name	num_shop
1	5	Galaxy S10	4
2	6	Galaxy S20	4
3	9	Samsung 55 Inches TV	3
4	10	Samsung 22 Inches TV	3

c) Excel result

id	name	no. shops
5	Galaxy S10	4
6	Galaxy S20	4
9	Samsung 55 Inches TV	3
10	Samsung 22 Inches TV	3

6. Find shops that made the most revenue in August 2020

a) SQL query

```
SELECT TOP (1) WITH TIES
      S.shop_name, PIO.shop_id, SUM(PIO.price * PIO.quantity) AS total_revenue
FROM shops S, products_in_order PIO, orders O
WHERE MONTH(O.ordered_date) = 8 AND YEAR(O.ordered_date) = 2020
      AND PIO.order_id = O.order_id
      AND PIO.shop_id = S.shop_id
GROUP BY S.shop_name, PIO.shop_id
ORDER BY total_revenue DESC;
```

b) SQL result

	shop_name	shop_id	total_revenue
1	Harvey Norman	9	811116.00

c) Excel result

shop id	name	revenue
9	Harvey Norman	811116

7. For users that made the most amount of complaints, find the most expensive products he/she has ever purchased.

a) SQL query

```
SELECT TOP (1) WITH TIES
      U.user_id, U.name, COUNT(C.complaint_id) as num_of_complaints
INTO #max_comp
FROM users U, complaints C
WHERE U.user_id = C.user_id
GROUP BY U.user_id, U.name
ORDER BY num_of_complaints DESC;

SELECT MC.user_id, MC.name, P.product_name, PIO.product_id, PIO.price
INTO #all_prod
FROM #max_comp MC, products P, products_in_order PIO, orders O
WHERE MC.user_id = O.user_id
      AND PIO.order_id = O.order_id
      AND PIO.product_id = P.product_id;

SELECT AP.user_id, AP.name, AP.product_name, AP.product_id, AP.price
FROM #all_prod AP
WHERE AP.price = (SELECT MAX(AP1.price)
                  FROM #all_prod AP1
                  WHERE AP.user_id = AP1.user_id);

DROP TABLE #max_comp;
DROP TABLE #all_prod;
```

b) SQL result

	user_id	name	product_name	product_id	price
1	525	Mia	iPhone 12	1	1150.00
2	56	Garval	iPhone 12	1	1240.00

c) Excel result (note that shop_prod merges both shop_id and product_id, just for debugging purposes)

user id	no. complaints	shop_prod	price
56	5	9_01	1240
525	5	5_01	1150

8. Find products that have never been purchased by some users, but are the top 5 most purchased products by other users in August 2020.

a) SQL query

```
SELECT DISTINCT product_id
INTO #products_not_all_purchased
FROM products_in_order PIO
WHERE EXISTS(SELECT U.user_id
             FROM users U
             WHERE NOT EXISTS (SELECT *
                               FROM products_in_order PIO1, orders O
                               WHERE PIO.product_id = PIO1.product_id
                               AND O.user_id = U.user_id
                               AND O.order_id = PIO1.order_id));

SELECT TOP (5) WITH TIES PIO.product_id, SUM(quantity) AS total_quantity
FROM products_in_order PIO, orders O
WHERE MONTH(ordered_date) = 8 AND YEAR(ordered_date) = 2020
      AND PIO.product_id IN (SELECT product_id
                             FROM #products_not_all_purchased)
      AND O.order_id = PIO.order_id
GROUP BY PIO.product_id
ORDER BY total_quantity;

DROP TABLE #products_not_all_purchased;
```

b) SQL result

	product_id	total_quantity
1	17	202
2	6	203
3	1	214
4	34	217
5	5	217

9. Find products that are increasingly being purchased over at least 3 months.

Note that the current month is October, and the query below checks for products that are increasingly being purchased from July to September (there are no October sales).

a) SQL query

```
SELECT DISTINCT PIO.product_id
FROM products_in_order PIO
WHERE (SELECT SUM(PIO1.quantity)
       FROM products_in_order PIO1, orders O1
       WHERE MONTH(O1.ordered_date) = MONTH(GETDATE()) - 1
             AND YEAR(O1.ordered_date) = YEAR(GETDATE())
             AND PIO1.order_id = O1.order_id
             AND PIO1.product_id = PIO.product_id)
       >
       (SELECT SUM(PIO2.quantity)
        FROM products_in_order PIO2, orders O2
        WHERE MONTH(O2.ordered_date) = MONTH(GETDATE()) - 2
              AND YEAR(O2.ordered_date) = YEAR(GETDATE())
              AND PIO2.order_id = O2.order_id
              AND PIO2.product_id = PIO.product_id)
       AND
       (SELECT SUM(PIO3.quantity)
        FROM products_in_order PIO3, orders O3
        WHERE MONTH(O3.ordered_date) = MONTH(GETDATE()) - 2
              AND YEAR(O3.ordered_date) = YEAR(GETDATE())
              AND PIO3.order_id = O3.order_id
              AND PIO3.product_id = PIO.product_id)
       >
       (SELECT SUM(PIO4.quantity)
        FROM products_in_order PIO4, orders O4
        WHERE MONTH(O4.ordered_date) = MONTH(GETDATE()) - 3
              AND YEAR(O4.ordered_date) = YEAR(GETDATE())
              AND PIO4.order_id = O4.order_id
              AND PIO4.product_id = PIO.product_id);
```

b) SQL result (left) and Excel result (right)

	product_id
1	1
2	2
3	4
4	5
5	7
6	10
7	13
8	14
9	15
10	18
11	19
12	21
13	23
14	25
15	26
16	27
17	29
18	32
19	35

prod id	jul < aug < sep
1	TRUE
2	TRUE
4	TRUE
5	TRUE
7	TRUE
10	TRUE
13	TRUE
14	TRUE
15	TRUE
18	TRUE
19	TRUE
21	TRUE
23	TRUE
25	TRUE
26	TRUE
27	TRUE
29	TRUE
32	TRUE
35	TRUE