# CS348 - Fall 2018 - Project 2
# PL/SQL

Due: Tuesday, November 6, 2018 at 11:59PM on Blackboard
(There will be a 10% penalty for each late calendar day. After five calendar days, the homework will not be accepted.)

In this project, you are asked to complete 5 procedures with PL/SQL. PL/SQL is only covered in the PSOs not in the class lectures.

Notes:

1. The schema definition of database tables and sample test data are provided in **tables.sql** and **data.sql** respectively. You need to use **droptables.sql** to clean your database before you start this project because test data may be different from the data used in project 1.
2. You should finish all your work in **answer.sql**. Skeleton code for procedures is already provided in answer.sql. Oracle will give error messages if you don't finish all of them, so you can comment the unfinished ones during development.
3. Please, don't change the names of procedures in answer.sql.
4. You can assume that all possible input to procedures 1-4 will be legal input, but we will test illegal input in procedure 5. So, the exception block is not required for testing procedures 1-4. Please, refer to the requirement of procedure 5 for details.
5. Submit your answer via **BlackBoard**.
6. Hints: you may want to use the command "**show errors;**" to debug your procedures.

The detailed requirements of each procedure are listed below:

1. **RetailerDetail**

   Create a procedure that shows the detailed information of a specific retailer, given the RetailerId as input. The detailed information should include the following:
   a. Retailer Name
   b. Retailer Address
   c. Retailer total orders in Orders table.
   d. Most popular product and the amount sold in the Orders table. We define the most popular product as the one having the highest number of items sold in the Orders table. *You can assume there is only one most popular product*.

   <u>The output should be in the following format</u>:

```
Retailer Name: Retailer A
Retailer Address: r_ddress1
Retailer Total Orders: 7
Most Popular Product: inferno
```

```
Total Sold: 115
```

2. **MonthlyDelayReport**

   Create a procedure that can generate a simple report for the delayed orders of all months in the database. For each month, you need to list the total number of delayed orders and the retailers that have delayed orders in that month. Only display the months with delayed orders. The ordering of months should be from the earliest to the lastest, while the ordering of retailers should be by retailer's name.

   The output should be in the following format:

```
Delayed orders in 2018-1: 2
Retailers with delayed orders:
- Retailer A
Delayed orders in 2018-2: 2
Retailers with delayed orders:
- Retailer A
- Retailer B
Delayed orders in 2018-5: 1
Retailers with delayed orders:
- Retailer A
Delayed orders in 2018-6: 1
Retailers with delayed orders:
- Retailer B
```

3. **LeastProfitProduct**

   Create a procedure that can generate a simple report for the products with least average profit in each category. Here, we define the profit of a product as the average of (Orders.UnitPrice - Products.ExfactoryPrice), given the product appears in different orders. Notice that there may be **more than one product** with the same average profit. Display all the products names and their profit for all categories.

   The output should be in the following format:

```
Least Profit in electronics
- Pixel Book: 75
Least Profit in books
- Foundation of Crypto: 30
- da vinci code: 30
Least Profit in apparel
- Adidas cap: 11
Least Profit in text_books
- inferno: 30
```

4. **RetailerProductCatergory**

   Create a table called RetailerCatergoryTable, with the most recent information about how many orders of each retailer are for products in the categories: electronic, apparel and books respectively.

   The output after running “select * from RetailerCatergoryTable;” should look like:

```
RETAILERID ELECTRONIC    APPAREL      BOOKS
---------- ---------- ---------- ----------
         1          1          1          2
         2          4          0          0
         3          0          1          0
         4          1          0          0
         5          1          0          0
```

5. **CustomerProductInfo**

   Given a customer id and a product id as input, list **all** order dates in which the customer orders the product. You may need to use **Exception** in PL/SQL to prevent your procedure from crashing if given an invalid customer id or an invalid product id.

   The output of a valid input should be similar to:

```
Records of customer id 1 with product id 1:
OrderDate: 01-JAN-18
```

   The output of invalid inputs should be similar to:

```
Records of customer id -1 with product id 1:
Invalid customer id or product id!

Records of customer id 1 with product id -1:
Invalid customer id or product id!

Records of customer id 100 with product id 100:
Invalid customer id or product id!
```

   (If there is no customer id 100 or product id 100 in database)