ATAK
COSMETICS

Database Design Project

Team 7 - IEOR 115 Fall 2021

Agenda

1. Client Overview
2. EER Diagram
3. Relations
4. SQL
5. Queries
6. Normalization
7. Conclusion

Meet the Team!

Erel Saul, Client Liaison
IEOR & BA

Kaan Gezguc, CEO
IEOR

Yash Bhandari, CCO
IEOR

Nicole Guzhavin
IEOR

Nikita Boddu
IEOR

Jonathan Brian
IEOR

Begum Dogan
ORMS

Carrie Liu
IEOR

Majd Muna
IEOR

Jillian Criscuolo
DS

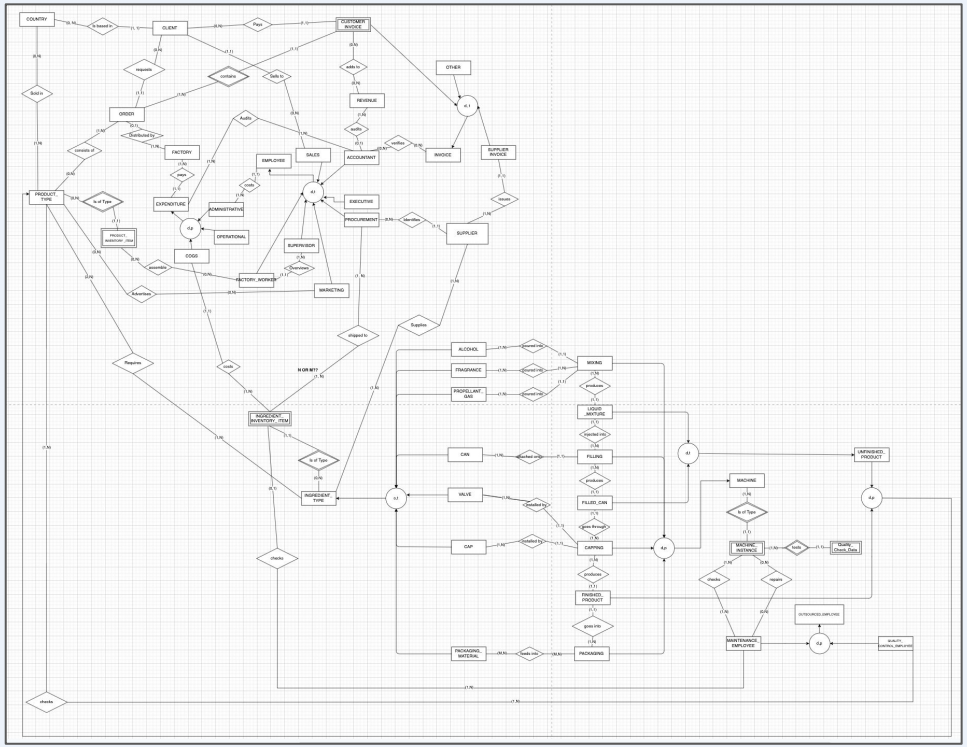
Noah Kaminer
Econ & DS

Meet ATAK Cosmetics

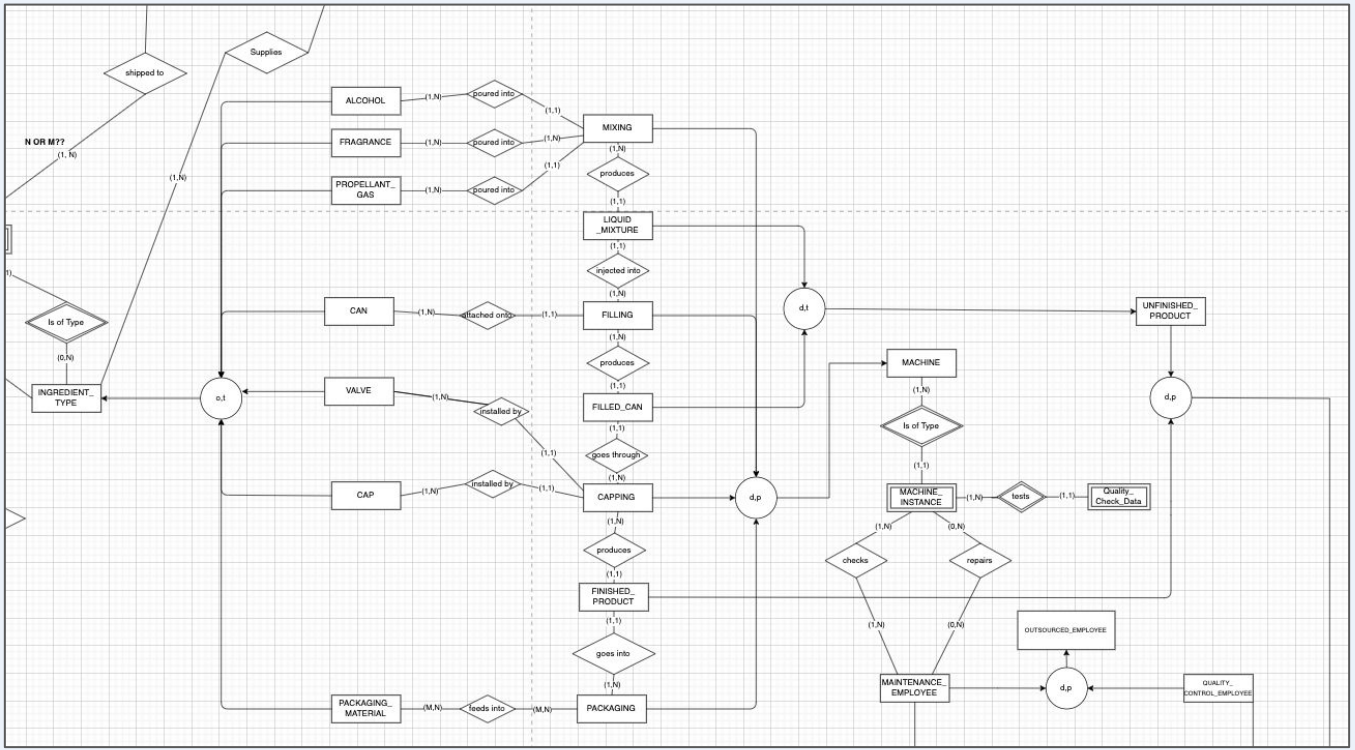
- ATAK Cosmetics – international manufacturer and exporter of cosmetics based in Istanbul, Turkey
  - Has around 230 employees and exports to over 50 countries
- Currently stores their data in Microsoft Excel files
  - Keeps track of agreements, pricing sheets, manufacturing and product details, and customer relationship management information

Our Goal → create a mySQL database that keeps track of ATAK's manufacturing processes, product assembly, and quality assurance checks from employee to financial records

EER Diagram


COUNTRY
CLIENT
CUSTOMER'S INVOICE
OTHER
REVENUE
GRIDDY
FACTORY
EMPLOYEE
SALES
ACCOUNTANT
INVOICE
SUPPLIER INVOICE
PRODUCT_TYPE
EXECUTIVE
PROCUREMENT
SUPPLIES
ADMINISTRATIVE
OPERATIONAL
SUPERVISOR
COGS
FACTORY_WORKER
MARKETING
ALCOHOL
FRAGRANCE
PROPELLANT_GAS
MIXING
LIQUID_MIXTURE
CAN
VALVE
CAP
FILLING
FILLED_CAN
CAPPING
FINISHED_PRODUCT
PACKAGING_MATERIAL
PACKAGING
INGREDIENT_INVENTORY_ITEM
INGREDIENT TYPE
UNFINISHED_PRODUCT
MACHINE
MACHINE_MAINTENANCE
MAINTENANCE EMPLOYEE
OUTSOURCED_EMPLOYEE
QUALITY_CONTROL_EMPLOYEE

EER Diagram Cont.

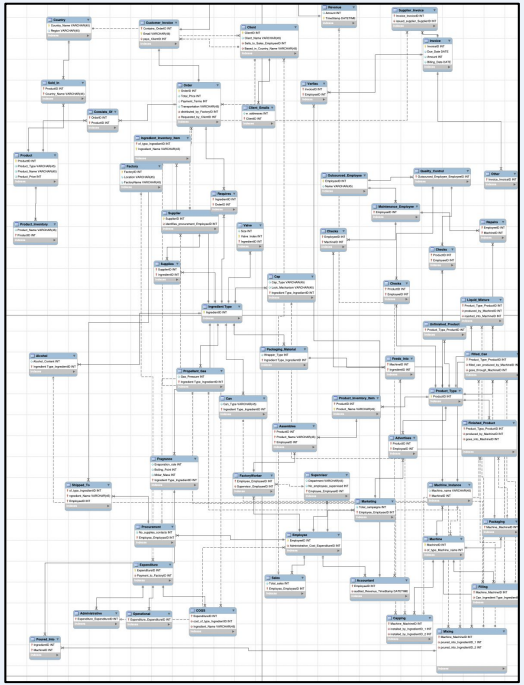


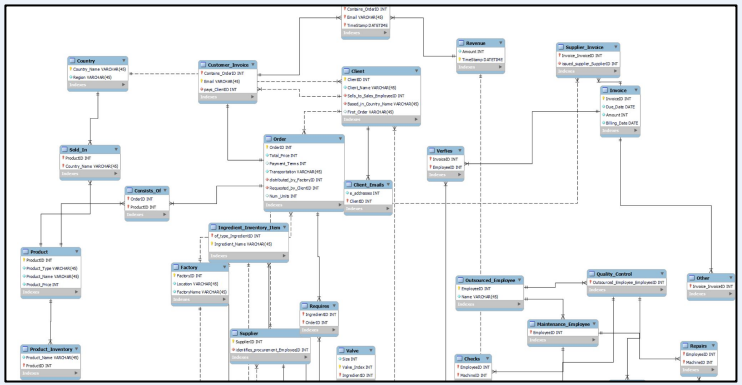
Relational Schema

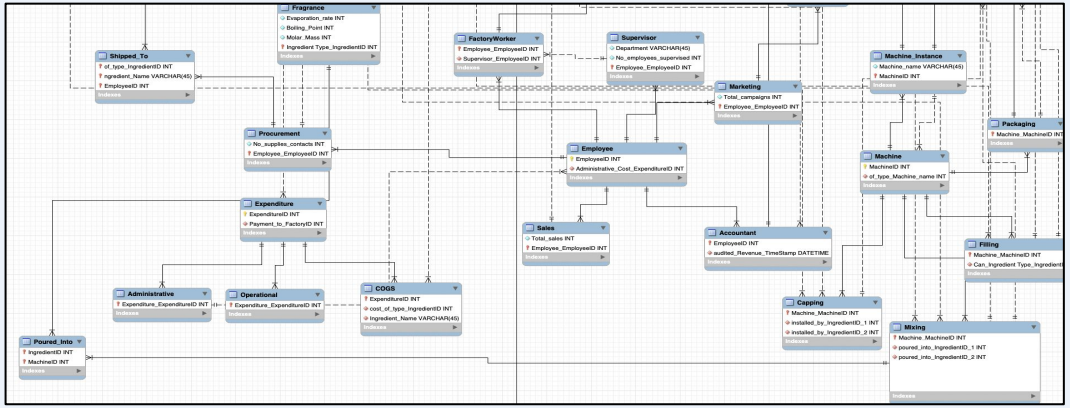
1. Country(Country_Name, Region)
2. Client(CustomerID, Client_Name, First_Order, sells_to_Sales_EmployeeID[6a], based_in_Country_Name[1])
3. Order(OrderID, Total_Price, Payment_Terms, Transportation, Date, Num_Units, distributed_by_FactoryID[4], requested_by_ClientID[2])
4. Factory(FactoryID, Factory_Name, Location)
5. Expenditure(ExpenditureID, payment_to_factory[4])
   a. Administrative(ExpenditureID[4])
   b. Operational(ExpenditureID[4])
6. Employee(EmployeeID, administrative_cost_EmployeeID[5a])
   a. Sales(EmployeeID[4], Total_sales)
   b. Accountant(EmployeeID[4], audited_revenue_timestamp[7])
   c. Procurement(EmployeeID[4], No_supplier_contacts)
   d. Marketing(EmployeeID[4], Total_campaigns)
   e. Supervisor(EmployeeID[4], Department, No_employees_supervised)
   f. Factory_Worker(EmployeeID[4], overview_supervisor_EmployeeID[6f])
7. Revenue(Timestamp, Amount)
8. Invoice(InvoiceID, Amount, Due_Date, Billing_Date)
   a. Supplier_Invoice(InvoiceID[8], issued_supplier_SupplierID[9])
   b. Other(InvoiceID[8])
9. Supplier(SupplierID, identifies_procurement_EmployeeID[6])
10. Outsourced_Employee(EmployeeID, name)
    a. Maintenance_Employee(EmployeeID[10])
    b. Quality_Control_Employee(EmployeeID[10])
11. Machine(Machine_Name)
    a. Mixing(Machine_Name[11], poured_into_Ingredient_Name_1[13a], poured_into_Ingredient_Name_2[13c])
    b. Capping(Machine_Name[11], installed_by_Ingredient_Name_1[13e], installed_by_Ingredient_Name_2[13f])
    c. Filling(Machine_Name[11], attached_onto_Ingredient_Name[14d])
    d. Packaging(Machine_Name[11])
12. Product_Type(Product_Name, Product_Price)
    a. Unfinished_Product(Product_Name[12])
       i. Liquid_Mixure(Product_Name[12], produced_by_Machine_Name[11a], injected_into_Machine_Name[11c])
       ii. Filled_Can(Product_Name[12], goes_through_Machine_Name[11b], filled_can_produced_by_Machine_Name[11c])
    b. Finished_Product(Product_Name[12], produced_by_Machine_Name[11a], goes_into_Machine_Name[11d])

13. Ingredient_Type(Ingredient_Name)
    a. Alcohol(Ingredient_Name[13], Alcohol_content)
    b. Fragrance(Ingredient_Name[13], Evaporation_rate, Boiling_point, Molar_mass)
    c. Propellant_Gas(Ingredient_Name[13], Gas_pressure)
    d. Can(Ingredient_Name[13], Can_type)
    e. Valve(Ingredient_Name[13], Valve_index, Size)
    f. Cap(Ingredient_Name[13], Cap_type, Lock_mechanism)
    g. Packaging_Material(Ingredient_Name[13], Wrapper_type)
14. Customer_Invoice(InvoiceID[8], Email, Date, paid_by_ClientID[2], contains_OID[3])
15. Product_Inventory_Item(Product_Name[12], Product_Inventory_Number)
16. Ingredient_Inventory_Item(Ingredient_Name[13], Inventory_Number, Quantity)
17. Machine_instance(MachineID, Machine_name[11])
18. Quality_Check(MachineID[17], Machine_name[11], FailureID , Datetime, Fail_Type, Time_Since_Last_Failure)
19. Sold_in(Country_Name[1], ProductID[12])
20. Consists_of(OrderID[3], ProductID[12b])
21. Requires(ProductID[12a], Ingredient_Name[13], Ingredient_Quantity)
22. Advertises(ProductID[12], EmployeeID[6])
23. Assembles(ProductID[12], Product_name[15], EmployeeID[6])
24. Checks(ProductID[12], EmployeeID[10b])
25. Adds_to(Timestamp[7], OrderID[3], Email[14])
26. Verifies(EmployeeID[6b], InvoiceID[8])
27. Supplies(Ingredient_Name[13], SupplierID[9])
28. Poured_into(Ingredient_Name[13b], Machine_Name[11a])
29. Feeds_into(Ingredient_Name[13e], Machine_Name[11d])
30. Checks(MachineID[17], Machine_Name[11], EmployeeID[10a])
31. Repairs(MachineID[17], Machine_Name[11], EmployeeID[10a])
32. Client_email(CustomerID[2], e_addresses)
33. Order_details(OrderID[3], ProductIDs)

SQL







Query 1 - Predicting Monthly Units Sold

- ATAK Use Case:
  - Optimize the raw ingredient sourcing process & better manage current inventory stockpiles
  - Can launch pinpointed marketing efforts to boost sales in specific months
  - Forecast revenue projections for improved capital allocation

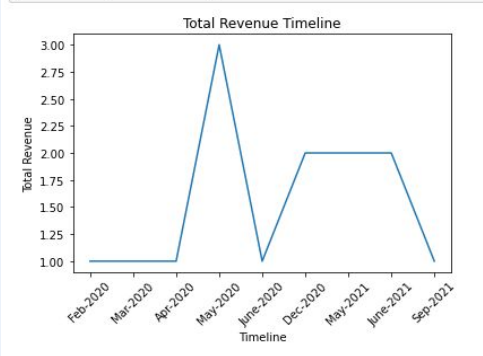
- SQL Query:
  - Data inserted into Client and Order tables via Python code from Lab 10

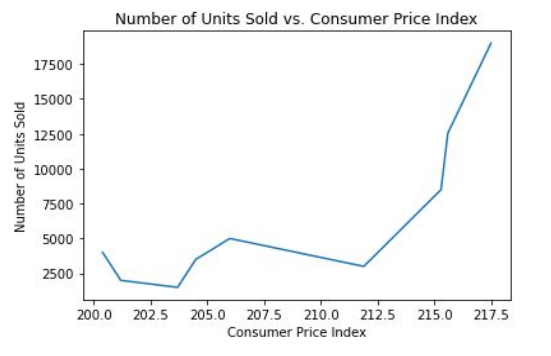
```sql
SELECT c.First_Order_Date,
       COUNT(o.OrderID) as Num_Orders,
       SUM(o.Total_Price) as Total_Revenue,
       SUM(o.Num_Units) as Num_Units_Sold,
       c.Num_New_Clients
FROM `Order` o, (SELECT COUNT(*) as Num_New_Clients,
                        DATE_FORMAT(First_Order, '%m%y') as First_Order_Date
                 FROM `Client`
                 GROUP BY First_Order_Date
                 HAVING First_Order_date IS NOT NULL) as c
WHERE DATE_FORMAT(o.Date, '%m%y') = c.First_Order_Date
GROUP BY c.First_Order_Date
ORDER BY c.First_Order_Date;
```

- Method:
  - Use the statsmodel package in Python to perform Linear Regression to predict monthly units sold
  - Independent Variables: Previous Month's Revenue, # of Orders, New Customers, Month, and CPI

Query 1 - Units Sold & Revenue Trend Results

First_Order_Date	Num_Orders	Total_Revenue	Num_Units_Sold	Num_New_Clients
0220	1	8387	3500	1
0320	1	6144	4000	1
0420	1	4215	2000	1
0520	3	12501	12550	1
0521	2	9446	3000	2
0620	1	5162	1500	1
0621	2	14598	19000	1
0921	1	7322	5000	1
1120	1	8223	NULL	1
1220	2	12052	8500	2





Query 1 - Linear Regression Model Results

	First_Order_Date	Num_Orders	Total_Revenue	Num_Units_Sold	Num_New_Clients	Month	Year	CPI	Month-Year
0	220	1	8387	3500.0	1	Feb	2020	204.5	Feb-2020
1	320	1	6144	4000.0	1	Mar	2020	200.4	Mar-2020
2	420	1	4215	2000.0	1	Apr	2020	201.2	Apr-2020
3	520	3	12501	12550.0	1	May	2020	215.6	May-2020
4	521	2	9446	3000.0	2	May	2021	211.9	May-2021
5	620	1	5162	1500.0	1	Jun	2020	203.7	June-2020
6	621	2	14598	19000.0	1	Jun	2021	217.5	June-2021
7	921	1	7322	5000.0	1	Sep	2021	206.0	Sep-2021
9	1220	2	12052	8500.0	2	Dec	2020	215.3	Dec-2020

We can predict the number of units sold in a given month and year based on this equation. We input the number of new clients, orders, indicators, of the month/year we are in, CPI of the month, and the total revenue.

For example, we are in December 2021, have 2 clients, 1 order, total revenue of 14562, and the CPI is 213.5, the number of units sold is predicted to be 4972 as shown below.

The model tells us:

Num_Units_Sold = 10.19 - 5190(Num_New_Clients) + 830(Num_Orders) + 1.62(Total_Revenue) + -2.39(CPI) + -1807(T.Dec) - 5253(T.Feb) - 2029(T.June) - 1128(T.Mar) - 4503(T.May) - 3435(T.Sep) + 1412(T.2021)

```
int(model.predict(pd.DataFrame({"Num_New_Clients": [2],"Month":['Dec'],"Year": ['2021'],"CPI": [213.5],"Total_Revenue":[9514],"Nu
```

```
4972
```

```
import statsmodels.formula.api as smf

ols1 = smf.ols(formula='Num_Units_Sold ~ Num_Orders + Num_New_Clients + Month + Total_R
                data=df)
model = ols1.fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          Num_Units_Sold   R-squared:                       1.000
Model:                             OLS   Adj. R-squared:                    nan
Method:                  Least Squares   F-statistic:                       nan
Date:                 Thu, 09 Dec 2021   Prob (F-statistic):                nan
Time:                         12:14:03   Log-Likelihood:                 175.69
No. Observations:                    9   AIC:                            -333.4
Df Residuals:                        0   BIC:                            -331.6
Df Model:                            8
Covariance Type:             nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      10.1864        inf          0        nan         nan         nan
Month[T.Dec] -1806.7741        inf         -0        nan         nan         nan
Month[T.Feb] -5253.0998        inf         -0        nan         nan         nan
Month[T.Jun] -2028.6966        inf         -0        nan         nan         nan
Month[T.Mar] -1127.9785        inf         -0        nan         nan         nan
Month[T.May] -4503.2354        inf         -0        nan         nan         nan
Month[T.Sep] -3435.1347        inf         -0        nan         nan         nan
Year[T.2021]  1411.5204        inf          0        nan         nan         nan
Num_Orders     829.8375        inf          0        nan         nan         nan
Num_New_Clients -5189.7081     inf         -0        nan         nan         nan
Total_Revenue     1.6206       inf          0        nan         nan         nan
CPI              -2.3906       inf         -0        nan         nan         nan
```

Query 2 - Frequency of Machine Instance Failures

- ATAK Use Case:
  - Estimate time between machine failures to create a better maintenance schedule
  - Understand what types of problems are causing machine failures most frequently
  - Reduce machine downtime and improve operational efficiency
- SQL Query:
  - Data inserted into Quality Check tables via Python code from Lab 10

```sql
SELECT q.Machine_Instance_MachineID, q.Machine_Instance_Machine_name, q.Fail_Type,
       COUNT(q.Machine_Instance_MachineID) as Frequency_of_Failure_Type,
       AVG(q.Time_Since_Last_Failure) as Average_Time_Between_Failures
FROM `Quality Check` q
GROUP BY q.Machine_Instance_MachineID, q.Machine_Instance_Machine_name, q.Fail_Type;
```

- Method:
  - Model failure times as a poisson process to estimate the likelihood of a failure at a certain time

Query 2 - Machine Breakdown Results

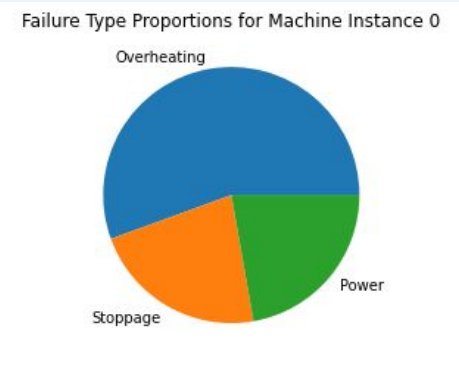
Machine_Instance_Ma...	Machine_Instance_Machine_na...	Fail_Type	Frequency_of_Failure_Ty...	Average_Time_Between_Failur...
0	Filling	Overheating	5	67.6
0	Filling	Stoppage	2	99
0	Filling	Power	2	77
1	Capping	Power	1	281
1	Capping	Stoppage	3	143
2	Packaging	Overheating	2	17
2	Packaging	Power	3	166.33333333333334
3	Mixing	Overheating	1	11
3	Mixing	Stoppage	1	0

Here, we can see how frequent each machine fails, and the total average time spent between failures for each respective failure type. We can tell that Machine_Instance_MachineID 0 has the most failures, which is an area of concern that ATAK can look further into.

```
In [10]: df.groupby(['Machine_Instance_MachineID']).sum()
```

Out[10]:

Machine_Instance_MachineID	Frequency_of_Failure_Type	Average_Time_Between_Failures
0	9	243.600000
1	4	424.000000
2	5	183.333333
3	2	11.000000



Query 2 - Poisson Distribution for Fail Time Results

Machine_Instance_MachineID 0 frequently overheats, showing 5 occurrences with an average time of 67 between failures. We use stochasticity and a poisson process to model the probability distribution for overheat failure per time for this machine. We notice that most failures will likely be between 57 and 77 days (flexion point)

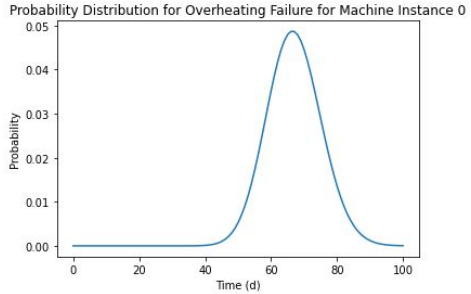
```
from scipy.stats import poisson

storage =[]

for i in np.linspace(0,100,101):
    storage = np.append(storage,poisson.pmf(i,67))

storage;
```

```
plt.plot(np.linspace(0,100,101), storage)
plt.title('Probability Distribution for Overheating Failure for Machine Instance 0')
plt.xlabel('Time (d)')
plt.ylabel('Probability');
```



Normalization Analysis: Relation 1

Quality_Check (MachineID, Machine_Name, FailureID, Datetime, Weekday, Fail_Type, Time_Since_Last_Failure, Machine_Start_Date, Components_Fixed)

- Components_Fixed is a multi-valued attribute - Quality_Check is NOT in 1NF.

Functional Dependencies:

{MachineID, MachineName, FailureID} -> {Datetime, Weekday, Fail_Type, Time_Since_Last_Failure, Components_Fixed}

{Datetime} -> {Weekday}

{MachineID, Machine_Name} -> {Machine_Start_Date}

Normalized to BCNF:

Quality_Check_Machine_Fix (MachineID, Machine_Name, FailureID, Datetime, Fail_Type, Time_Since_Last_Failure)

Machine_Lifetime(MachineID, Machine_Name, Machine_Start_Date)

Components_Fixed(MachineID, Machine_Name, FailureID, Components_Fixed)

Date_Day (Datetime, Weekday)

Normalization Analysis: Relation 2

Customer_Invoice (InvoiceID, Email, Date, ClientID, OrderID)

- There are no multi-valued attributes, but not all non-prime attributes solely depend on the whole key
- In Customer_Invoice, ClientID is fully functionally dependent on Email - Customer_Invoice is NOT in 2NF

Functional Dependencies:

{InvoiceID, Email} -> {Date, OrderID}

{Email} -> {ClientID}

Normalized to BCNF:

Invoice_Instance (InvoiceID, Email, Date, OrderID)

Client_Info(Email, ClientID)

Normalization Analysis: Relation 3

Ingredient_Inventory_Item (Ingredient_Name, Inventory_Number, Quantity, Max_Inventory, Eligible_Storage_Space)

- Eligible_Storage_Space is a multi-valued attribute – Ingredient_Inventory_Item is NOT in 1NF.

Functional Dependencies:

{Ingredient_Name} -> {Max_Inventory}

{Ingredient_Name, Inventory_Number} -> {Quantity}

Normalized to BCNF:

Ingredient_Inventory_Info (Ingredient_Name, Max_Inventory)

Ingredient_Storage_Info (Ingredient_Name, Eligible_Storage_Space)

Ingredient_Instance (Ingredient_Name, Inventory_Number, Quantity)

Normalization Analysis: Relation 4

Client (ClientID, EmployeeID, Name, Country_Name, Date)

- There are no multi-valued attributes, but not all non-prime attributes solely depend on the whole key
- We would like to record each Client - Employee interaction, and each client can interact with a specific employee more than once

Functional Dependencies:

{ClientID} -> {Client_Name, Country_Name}

Normalized to BCNF:

Client_Detail (ClientID, Name, Country_Name)

Sales_Employee_Interaction (ClientID, EmployeeID, Date)

Conclusion

- Centralized mySQL database where ATAK can track and optimize all steps of the manufacturing and distribution process

- Accessible data for accurate modeling and prediction to address global supply chain crisis

- Unlimited future possibilities for the expansion of ATAK Cosmetics

ATAK
COSMETICS

Thank you!

Questions?