TP1-2018

May 3, 2018


```
In [1]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        import datetime as datetime
        import warnings
        warnings.filterwarnings('ignore')

        %matplotlib inline
```

1 CARGA DE DATOS

```
In [2]: postulantesEducacion=pd.read_csv('fiuba_1_postulantes_educacion.csv',low_memory=False)
        postulantesGeneroYEdad=pd.read_csv('fiuba_2_postulantes_genero_y_edad.csv',low_memory=Fa
        vistas=pd.read_csv('fiuba_3_vistas.csv',low_memory=False)
        postulaciones=pd.read_csv('fiuba_4_postulaciones.csv',low_memory=False)
        avisosOnline=pd.read_csv('fiuba_5_avisos_online.csv',low_memory=False)
        avisosDetalle=pd.read_csv('fiuba_6_avisos_detalle.csv',low_memory=False)
```

** Merge entre los archivos:** ==========

** 1. Educacion, genero y edad de los postulantes**

```
In [3]: postulantes = pd.merge(postulantesEducacion, postulantesGeneroYEdad, on='idpostulante',
        postulantes.tail()
```

```
Out[3]:        idpostulante nombre estado fechanacimiento        sexo
        310362      mzdNo99    NaN    NaN      1999-08-10         FEM
        310363      Nzr1J34    NaN    NaN             NaN  NO_DECLARA
        310364      JBrNMNk    NaN    NaN      1997-01-11        MASC
        310365      2zP6Jd0    NaN    NaN      1997-03-06         FEM
        310366      2zP6J9V    NaN    NaN             NaN  NO_DECLARA
```

2. AvisosOnline y avisosDetalle, asi generan un solo archivo de avisos

```
In [4]: avisos = pd.merge(avisosOnline, avisosDetalle, on='idaviso', how='outer')
        avisos.tail()
```

```
Out[4]:           idaviso  idpais                                              titulo  \
        13844  1112342204     1.0  Ejecutivo de Ventas - Prov. de Bs. As e Interi...
        13845  1112341862     1.0  24 Feb - Chief Software Architect ($100K/yr) -...
        13846  1112341899     1.0  24 Feb - Customer Support Manager ($100K/yr) -...
        13847  1112341992     1.0  24 Feb - VP of Customer Support ($200K/yr) - O...
        13848  1112342193     1.0                        Supervisor de Mantenimiento

                                                descripcion        nombre_zona  \
        13844  <p>Editorial Rubinzal Culzoni, líder en el ámb...  Gran Buenos Aires
        13845  <p>Para inscribirse en el torneo debe ingresar...  Gran Buenos Aires
        13846  <p>Para inscribirse en el torneo debe ingresar...  Gran Buenos Aires
        13847  <p>Para inscribirse en el torneo debe ingresar...  Gran Buenos Aires
        13848  <p>Nuestro cliente, un importante PYME dedicad...  Gran Buenos Aires

              ciudad mapacalle tipo_de_trabajo                    nivel_laboral  \
        13844    NaN       NaN       Full-time              Senior / Semi-Senior
        13845    NaN       NaN     Teletrabajo  Jefe / Supervisor / Responsable
        13846    NaN       NaN     Teletrabajo  Jefe / Supervisor / Responsable
        13847    NaN       NaN     Teletrabajo  Jefe / Supervisor / Responsable
        13848    NaN       NaN       Full-time  Jefe / Supervisor / Responsable

                        nombre_area denominacion_empresa
        13844                 Ventas   Rubinzal Editorial
        13845  Tecnologia / Sistemas            CrossOver
        13846        Soporte Técnico            CrossOver
        13847    Atención al Cliente            CrossOver
        13848          Mantenimiento        VF CONSULTING
```

2 VISUALIZACIONES

3 *1. Hora con mayor y menor cantidad de vistas*

- Verificacion de la calidad de datos

```
In [5]: vistas.dtypes

Out[5]: idAviso          int64
        timestamp       object
        idpostulante    object
        dtype: object

In [6]: vistas.timestamp.value_counts().head()

Out[6]: 2018-02-28T18:13:14.254-0500    3
        2018-02-27T10:14:18.766-0500    3
        2018-02-26T12:22:26.834-0500    3
        2018-02-24T09:49:12.766-0500    3
        2018-02-26T09:52:20.019-0500    3
        Name: timestamp, dtype: int64

In [7]: vistas.isnull().any()

Out[7]: idAviso         False
        timestamp       False
        idpostulante    False
        dtype: bool
```

- Se agregan la columnas dia, mes, año y hora

```
In [5]: # Convertimos año, mes, dia de la semana y hora
        import calendar
        vistas['timestamp'] = pd.to_datetime(vistas['timestamp'])
        vistas['Anio'] = vistas['timestamp'].map(lambda x:x.year)
        vistas['Mes'] = vistas['timestamp'].map(lambda x:x.month)
        vistas['Dia'] = vistas['timestamp'].map(lambda x:x.weekday_name)
        vistas['Hora'] = pd.to_datetime(vistas['timestamp'], format='%H:%M',errors='coerce').dt.

        vistas.rename(columns={'idAviso': 'idaviso'}, inplace=True)
```

- Despreciamos el mes de 3, ya que la cantidad de vistas son muy chicas a comparacion a las del mes 2

```
In [9]: vistas.Mes.value_counts()

Out[9]: 2    921074
        3     40823
        Name: Mes, dtype: int64

In [6]: vistasFebrero = vistas[vistas.Mes==2]
        vistasFebrero.tail()

Out[6]:            idaviso               timestamp idpostulante  Anio  Mes        Dia  \
        926019  1112352879 2018-02-28 23:59:43.373      vV9BGbE  2018    2  Wednesday
        926020  1112303807 2018-02-28 23:59:46.717      ZD8QEXE  2018    2  Wednesday
```

```
926021  1112322670 2018-02-28 23:59:50.148      bOj3ojq  2018    2  Wednesday
926022  1112347283 2018-02-28 23:59:53.949      ZDNJzJE  2018    2  Wednesday
926023  1112316657 2018-02-28 23:59:59.935      owavZOL  2018    2  Wednesday

        Hora
926019    23
926020    23
926021    23
926022    23
926023    23
```

- Eliminamos las columnas con la que no vamos a trabajar

```
In [11]: vistasFebrero.tail()

Out[11]:            idaviso               timestamp idpostulante  Anio  Mes        Dia  \
         926019  1112352879 2018-02-28 23:59:43.373      vV9BGbE  2018    2  Wednesday
         926020  1112303807 2018-02-28 23:59:46.717      ZD8QEXE  2018    2  Wednesday
         926021  1112322670 2018-02-28 23:59:50.148      bOj3ojq  2018    2  Wednesday
         926022  1112347283 2018-02-28 23:59:53.949      ZDNJzJE  2018    2  Wednesday
         926023  1112316657 2018-02-28 23:59:59.935      owavZOL  2018    2  Wednesday

                 Hora
         926019    23
         926020    23
         926021    23
         926022    23
         926023    23

In [12]: vistasFebrero.isnull().any()

Out[12]: idaviso         False
         timestamp       False
         idpostulante    False
         Anio            False
         Mes             False
         Dia             False
         Hora            False
         dtype: bool

In [8]: vistasFebreroPorHora=vistasFebrero

        grouped = vistasFebreroPorHora.loc[:,['Hora','idaviso']].groupby('Hora').agg(['count'])
        top_vistas = grouped[('idaviso','count')].sort_values(ascending=False)
        top_vistas.head()

Out[8]: Hora
        14    66168
        15    63737
```

```
13    60972
19    59214
20    58042
Name: (idaviso, count), dtype: int64
```

```
In [9]: vistasFebreroPorHora=vistasFebrero

        top_vistas = vistasFebreroPorHora.groupby('Hora').count().loc[:,'idaviso'].plot(linewidt

        plt.title('Cantidad de visitas según la hora', fontsize=18)
        top_vistas.set_ylabel('Cantidad de visitas', fontsize=10)
        top_vistas.set_xlabel('Horas', fontsize=10)
```

```
Out[9]: <matplotlib.text.Text at 0x7fcd6d1a97f0>
```

3.1 Conclusión:

3.1.1 Se registra que el horario con mayor visitas es a las 14 hs y el de menor visitas a las 8 hs.

4 *2.Día de la semana con mayor y menor cantidad de visitas*

```
In [11]: archivoVistas = vistas
         archivoVistas['timestamp'] = pd.to_datetime(archivoVistas['timestamp'])
         archivoVistas['timestamp_month'] = archivoVistas['timestamp'].dt.month
         archivoVistas['timestamp_weekday_name'] = archivoVistas['timestamp'].dt.weekday_name
```

In [12]:
```
vistasFebrero= archivoVistas[archivoVistas.timestamp_month==2]
```

In [13]:
```
selecc_vistas=vistasFebrero[['idaviso','timestamp_month','timestamp_weekday_name']]
selecc_vistas['pivot']=1
```

In [14]:
```
selecc_vistas=selecc_vistas.pivot_table(values='pivot',index='timestamp_weekday_name',c
selecc_vistas.columns=['FEBRERO']
selecc_vistas
```

Out[14]:

timestamp_weekday_name	FEBRERO
Friday	47236
Monday	227957
Saturday	95930
Sunday	90646
Tuesday	232145
Wednesday	227160

In [15]:
```
sorter = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
sorterIndex = dict(zip(sorter,range(len(sorter))))
sorterIndex
```

Out[15]:
```
{'Friday': 5,
 'Monday': 1,
 'Saturday': 6,
 'Sunday': 0,
 'Thursday': 4,
 'Tuesday': 2,
 'Wednesday': 3}
```

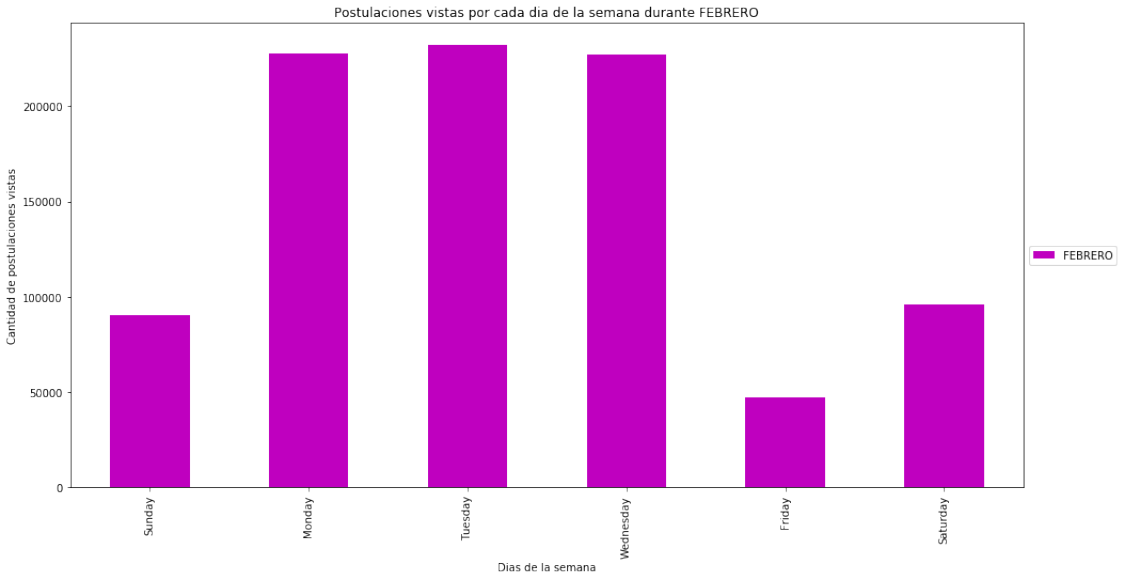
In [16]:
```
selecc_vistas['Day_id'] = selecc_vistas.index
selecc_vistas['Day_id'] = selecc_vistas['Day_id'].map(sorterIndex)
selecc_vistas=selecc_vistas.sort_values(by=['Day_id'],ascending=True)
selecc_vistas
```

Out[16]:

timestamp_weekday_name	FEBRERO	Day_id
Sunday	90646	0
Monday	227957	1
Tuesday	232145	2
Wednesday	227160	3
Friday	47236	5
Saturday	95930	6

In [17]:
```
selecc_vistas=selecc_vistas[['FEBRERO']]
f = plt.figure()
plt.title('Postulaciones vistas por cada dia de la semana durante FEBRERO', color='blac
ax = selecc_vistas.plot(kind='bar', stacked=False,figsize=(16,8),ax=f.gca(),color=['m']
handles, labels = ax.get_legend_handles_labels()
```

```
plt.legend(handles[::-1], labels[::-1],loc='center left', bbox_to_anchor=(1.0, 0.5))
ax.set_ylabel('Cantidad de postulaciones vistas');
ax.set_xlabel('Dias de la semana');
plt.show()
```



4.1 Conclusión:

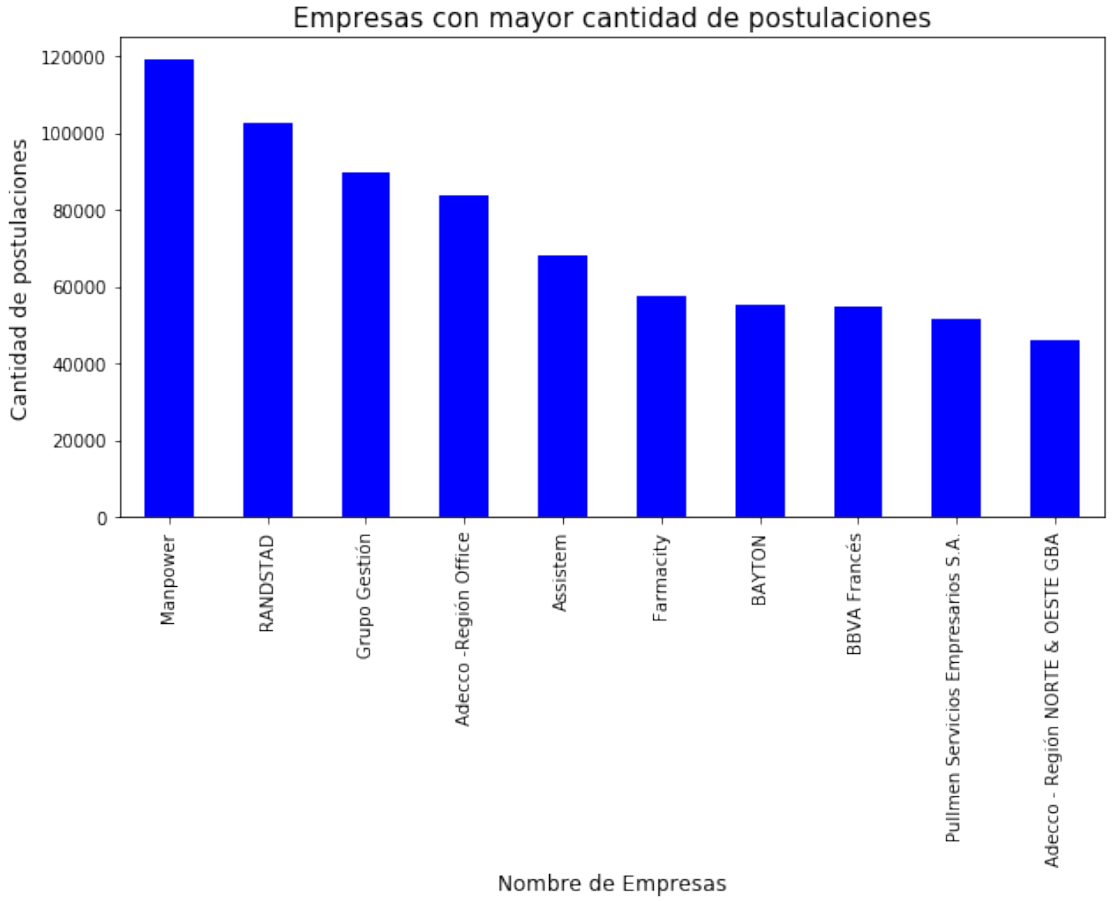
4.1.1 Se registra que el día con mayor visitas es el martes y el de menor es el viernes. De ello se puede decir que los postulantes estipulan que el día que lunes es cuando se publica la oferta laboral

5 *3.Empresas con mayor y menor cantidad de postulaciones*

```
In [22]: dataframe2= pd.merge(avisos,postulaciones, on='idaviso', how='inner')
         empresas=dataframe2[['idaviso','denominacion_empresa']]
         empresas.dropna(subset=['denominacion_empresa'],inplace=True)
         empresas.isnull().sum()

Out[22]: idaviso                0
         denominacion_empresa    0
         dtype: int64

In [23]: grouped = empresas.loc[:,['denominacion_empresa','idaviso']].groupby('denominacion_empr
         top10_empresas = grouped[('idaviso','count')].sort_values(ascending=False)
         top10_empresas.head()

Out[23]: denominacion_empresa
         Manpower                 119013
         RANDSTAD                 102640
```

```
Grupo Gestión              89950
Adecco -Región Office      83530
Assistem                   68125
Name: (idaviso, count), dtype: int64
```

```
In [24]: empresas.groupby('denominacion_empresa')['idaviso'].count().sort_values(ascending = Fal

         plt.title('Empresas con mayor cantidad de postulaciones', fontsize=15);
         plt.xlabel('Nombre de Empresas', fontsize=12);
         plt.ylabel('Cantidad de postulaciones', fontsize=12);
```



5.1 Conclusión:

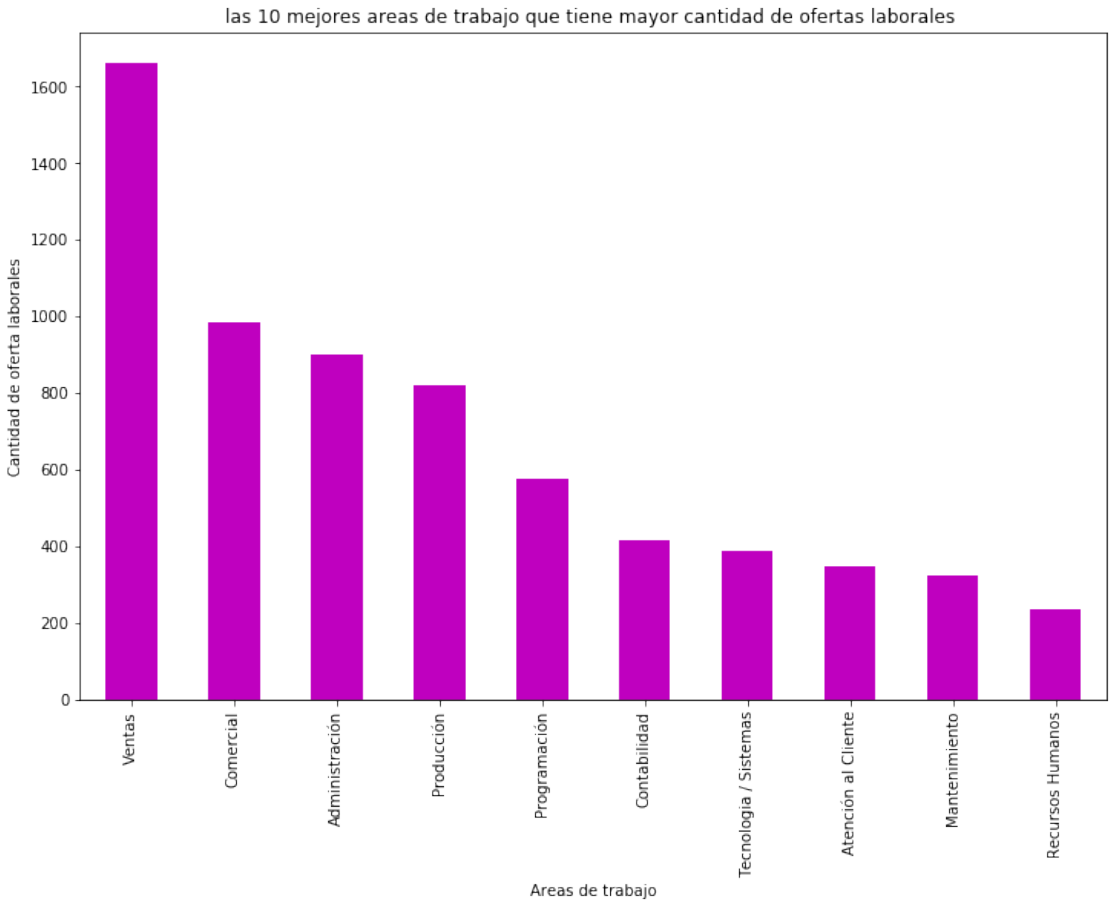
5.1.1 Se regista mayor postulaciones con Manpower, esto se debe a su amplia oferta laboral

4. Las 10 mejores areas que tienen la mayor cantidad de publicacion en ofertas laborales

==========

```
In [25]: f = plt.figure()
         avisos_areaTrabajo=avisos[['idaviso','nombre_area']]
```

```
grafhist = avisos_areaTrabajo['nombre_area'].value_counts().sort_values(ascending=False
#grafhist
plt.title('las 10 mejores areas de trabajo que tiene mayor cantidad de ofertas laborale

ax=grafhist.plot(kind='bar',rot=90,figsize=(12,8),color='m');
ax.set_ylabel('Cantidad de oferta laborales');
ax.set_xlabel('Areas de trabajo');
```



5.2 Conclusión:

5.2.1 Como podemos observar el area de Ventas, es la mayor area que ofrece a nuestros postulados una oportunidad de encontrar empleo,

6 5. Las 10 mejores areas que tienen mayor postulación

In [26]:
```
#verificamos que no existe ningun tipo de nan's
agrup_postulaciones=postulaciones [['idaviso','idpostulante']]
aviso_cant=agrup_postulaciones.groupby(['idaviso']).count().sort_values(by=['idpostulan
aviso_cant.rename(columns={'idpostulante':'cantPostulantes'},inplace=True)
aviso_cant.head(10)
```

```
Out[26]:       idaviso  cantPostulantes
          0  1112033906             9932
          1  1112334791             9787
          2  1112204682             9244
          3  1112094756             8763
          4  1112345900             8304
          5  1112319451             8025
          6  1112298966             7637
          7  1112262494             7541
          8  1112305277             7153
          9  1112296264             6968
```

```
In [27]: aviso_cant.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12692 entries, 0 to 12691
Data columns (total 2 columns):
idaviso            12692 non-null int64
cantPostulantes    12692 non-null int64
dtypes: int64(2)
memory usage: 198.4 KB
```

```
In [28]: aviso_area=avisos[['idaviso','nombre_area']]
         aviso_area.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13849 entries, 0 to 13848
Data columns (total 2 columns):
idaviso        13849 non-null int64
nombre_area    13534 non-null object
dtypes: int64(1), object(1)
memory usage: 324.6+ KB
```

```
In [29]: #eliminar los null
         aviso_area.dropna(inplace=True)
         aviso_area.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13534 entries, 0 to 13848
Data columns (total 2 columns):
idaviso        13534 non-null int64
nombre_area    13534 non-null object
dtypes: int64(1), object(1)
memory usage: 317.2+ KB
```

```
In [30]: result = pd.merge(aviso_cant,aviso_area,on='idaviso',how='outer')
         #quedo con las columnas de cantPostulantes y nombre_area
```

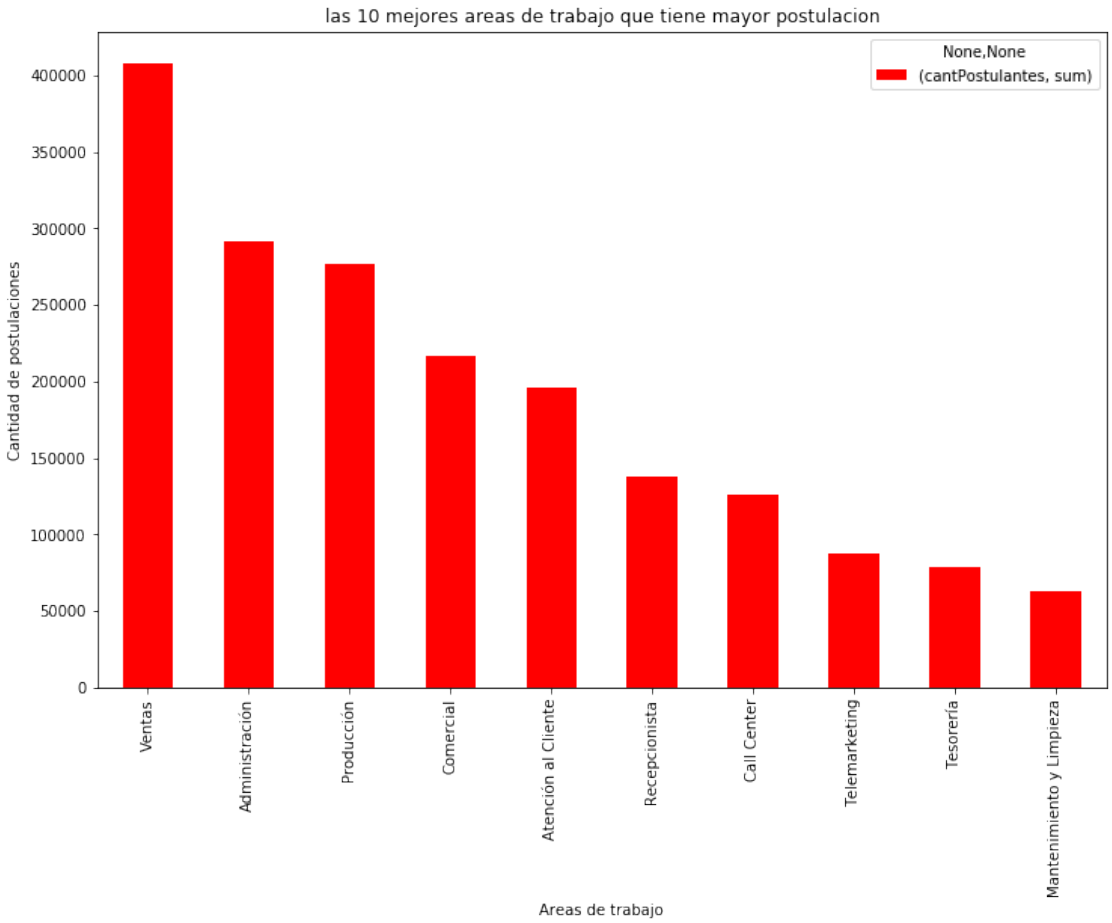
```
postulantes_area=result[['cantPostulantes','nombre_area']]
postulantes_area=postulantes_area.groupby('nombre_area').\
                    agg([np.sum]).sort_values(('cantPostulantes','sum'),ascending=False)

postulantes_area
```

Out[30]:

nombre_area	cantPostulantes sum
Ventas	408148.0
Administración	291135.0
Producción	277089.0
Comercial	216677.0
Atención al Cliente	195636.0
Recepcionista	137485.0
Call Center	126430.0
Telemarketing	87506.0
Tesorería	78450.0
Mantenimiento y Limpieza	63308.0

In [31]:
```
f= plt.figure()
graf = postulantes_area

ax=graf.plot(kind='bar',rot=90,figsize=(12,8),color='r');
ax.set_ylabel('Cantidad de postulaciones');
ax.set_xlabel('Areas de trabajo');
ax.set_title('las 10 mejores areas de trabajo que tiene mayor postulacion', color='blac
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



6.1 Conclusion:

6.1.1 Los postulantes, postularan a los puestos que mayor ofrecimiento laboral tengan y mayor sea la posibilidad de encontrae un puesto de trabajo

7 6. Genero de los postulantes para las distintas areas

```
In [32]: postulantesGeneroYEdad.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200888 entries, 0 to 200887
Data columns (total 3 columns):
idpostulante       200888 non-null object
fechanacimiento    196138 non-null object
sexo               200888 non-null object
dtypes: object(3)
memory usage: 4.6+ MB
```

```
In [33]: # me quedo con los id de los postulantes con sus respectivos generos
         postulante_genero=postulantesGeneroYEdad[['idpostulante','sexo']]
         postulante_aviso=postulaciones[['idaviso','idpostulante']]
         aviso_area=avisos[['idaviso','nombre_area']]

In [34]: agrupando_aviso= pd.merge(aviso_area,postulante_aviso,on='idaviso',how='outer')
         agrupando_aviso.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 3403927 entries, 0 to 3403926
Data columns (total 3 columns):
idaviso         int64
nombre_area     object
idpostulante    object
dtypes: int64(1), object(2)
memory usage: 103.9+ MB


In [35]: agrupando_postulante=pd.merge(agrupando_aviso,postulante_genero,on='idpostulante',how='
         agrupando_postulante.tail()

Out[35]:            idaviso             nombre_area idpostulante  sexo
         3403922  1112308336  Ingeniería de Producto      NzNaAll  MASC
         3403923  1112250944           Mantenimiento      NzNaAll  MASC
         3403924  1112261261           Mantenimiento      NzNaAll  MASC
         3403925  1112268019              Producción      NzNaAll  MASC
         3403926  1112310594                     NaN      NzNaAll  MASC

In [36]: agrupando_genero=agrupando_postulante[['nombre_area','sexo']]
         agrupando_genero.dropna(inplace=True)
         agrupando_genero.tail()

Out[36]:                    nombre_area  sexo
         3403921    Mineria/Petroleo/Gas  MASC
         3403922  Ingeniería de Producto  MASC
         3403923           Mantenimiento  MASC
         3403924           Mantenimiento  MASC
         3403925              Producción  MASC

In [37]: grupArea_genero=agrupando_genero
         grupArea_genero['pivot']=1
         grupArea_genero.tail()

Out[37]:                    nombre_area  sexo  pivot
         3403921    Mineria/Petroleo/Gas  MASC      1
         3403922  Ingeniería de Producto  MASC      1
         3403923           Mantenimiento  MASC      1
         3403924           Mantenimiento  MASC      1
         3403925              Producción  MASC      1
```

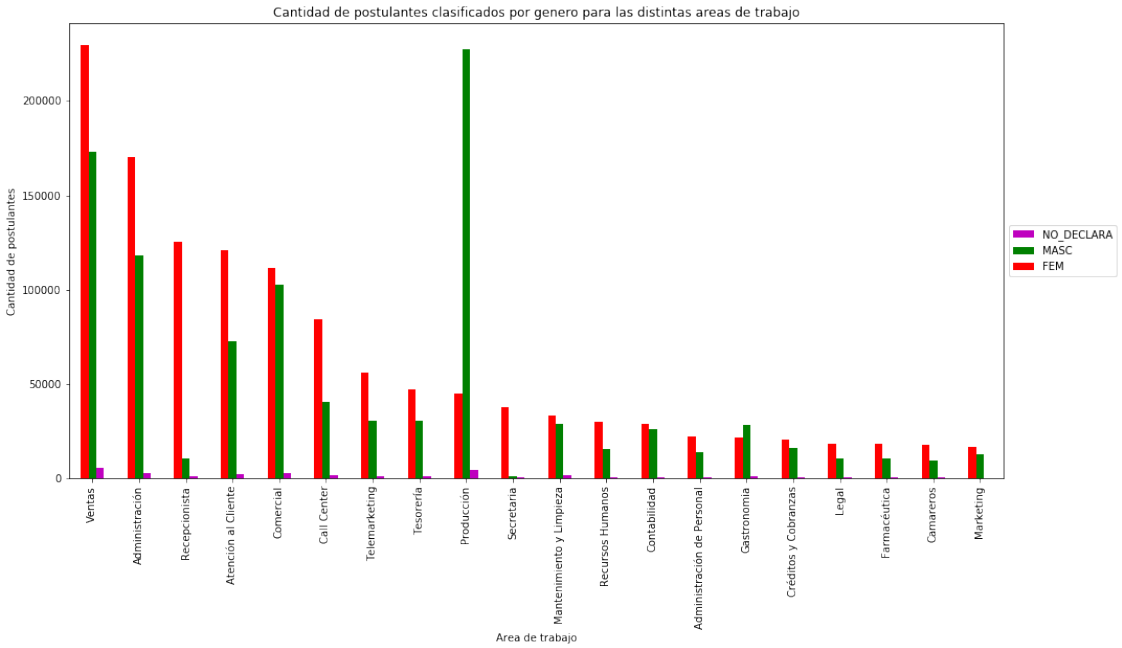
- Armo una tabla con los nombre de area y sus respectivos genero(femenino masculino y no_declara)

```
In [38]: grupArea_genero=grupArea_genero.pivot_table(values='pivot',index='nombre_area',columns=
         grupArea_genero=grupArea_genero.sort_values(by=['FEM','MASC','NO_DECLARA'],ascending=Fa
         grupArea_genero.head(10)
```

Out[38]:

sexo	FEM	MASC	NO_DECLARA
nombre_area			
Ventas	229674.0	172813.0	5661.0
Administración	170082.0	118377.0	2676.0
Recepcionista	125446.0	10716.0	1323.0
Atención al Cliente	120674.0	72786.0	2176.0
Comercial	111656.0	102438.0	2583.0
Call Center	84383.0	40505.0	1542.0
Telemarketing	55845.0	30455.0	1206.0
Tesorería	47126.0	30394.0	930.0
Producción	45077.0	227363.0	4649.0
Secretaria	37485.0	1393.0	310.0

```
In [39]: f = plt.figure()

         ax = grupArea_genero.plot(kind='bar', stacked=False,figsize=(16,8),ax=f.gca(), color=['
         handles, labels = ax.get_legend_handles_labels()
         plt.legend(handles[::-1], labels[::-1],loc='center left', bbox_to_anchor=(1.0, 0.5))
         ax.set_ylabel('Cantidad de postulantes');
         ax.set_xlabel('Area de trabajo');
         ax.set_title('Cantidad de postulantes clasificados por genero para las distintas areas
         plt.show()
```



7.1 Conclusión:

7.1.1 Podemos ver que la cantidad de genero no declarados son en definitiva despreciados en cantidad con respecto a los otros dos. Tambien podemos observar que en ventas tenemos la mayor cantidad de postulantes del género femenino. Lo que puede llamar la atencion es que en la parte gastronomica el genero masculino son los mas se postulan ya que antiguamente se creia que era un espacio solo para el genero femenino.

7.1.2 Ademas como vemos el genero masculino lidera y sobrepasa a la postulacion para el area de produccion. Veremos en la segunda parte de nuestro tp poder validar este grafico, ya que muchos aspectos sea por genero o por otras.

8 7. Postulaciones por cada dia de la semana durante ENERO y FEBRERO

```
In [10]: grup_postulaciones=pd.read_csv('/home/sherly/Escritorio/datos_navent_fiuba/fiuba_4_post
         grup_postulaciones['fechapostulacion']=pd.to_datetime(grup_postulaciones['fechapostulac
         grup_postulaciones['Year']=grup_postulaciones['fechapostulacion'].map(lambda x:x.year)
         grup_postulaciones['Year']=grup_postulaciones['Year'].fillna(0).astype(int)
         grup_postulaciones['Month']=grup_postulaciones['fechapostulacion'].map(lambda y:y.month
         grup_postulaciones['Month']=grup_postulaciones['Month'].fillna(0).astype(int)
         grup_postulaciones['Day']=grup_postulaciones['fechapostulacion'].map(lambda z:z.day)
         grup_postulaciones['Day']=grup_postulaciones['Day'].fillna(0).astype(int)
         grup_postulaciones.head()
```

Out[10]:

	idaviso	idpostulante	fechapostulacion	Year	Month	Day
0	1112257047	NM5M	2018-01-15 16:22:34	2018	1	15
1	1111920714	NM5M	2018-02-06 09:04:50	2018	2	6
2	1112346945	NM5M	2018-02-22 09:04:47	2018	2	22
3	1112345547	NM5M	2018-02-22 09:04:59	2018	2	22
4	1112237522	5awk	2018-01-25 18:55:03	2018	1	25

```
In [42]: grup_postulaciones.Month.unique()
```

```
Out[42]: array([1, 2])
```

```
In [43]: grup_postulaciones.Year.unique()
```

```
Out[43]: array([2018])
```

```
In [11]: # cambiamos los numeros de dias por el nombre del día que le corresponde
         import datetime, locale
         locale.setlocale(locale.LC_ALL, '')
         grup_postulaciones['Day of week']=grup_postulaciones['fechapostulacion'].dt.strftime("%
         grup_postulaciones.head()
```

Out[11]:

	idaviso	idpostulante	fechapostulacion	Year	Month	Day	Day of week
0	1112257047	NM5M	2018-01-15 16:22:34	2018	1	15	lunes
1	1111920714	NM5M	2018-02-06 09:04:50	2018	2	6	martes
2	1112346945	NM5M	2018-02-22 09:04:47	2018	2	22	jueves
3	1112345547	NM5M	2018-02-22 09:04:59	2018	2	22	jueves
4	1112237522	5awk	2018-01-25 18:55:03	2018	1	25	jueves

In [12]:
```
# me quedo con la columna idpostulante, mes, año y dia
selecc_postulaciones=grup_postulaciones[['idpostulante','Month','Day of week']]
selecc_postulaciones['pivot']=1
selecc_postulaciones.head()
```

Out[12]:

	idpostulante	Month	Day of week	pivot
0	NM5M	1	lunes	1
1	NM5M	2	martes	1
2	NM5M	2	jueves	1
3	NM5M	2	jueves	1
4	5awk	1	jueves	1

In [13]:
```
selecc_postulaciones=selecc_postulaciones.pivot_table(values='pivot',index='Day of week
selecc_postulaciones.columns=['ENERO','FEBRERO']
selecc_postulaciones
```

Out[13]:

Day of week	ENERO	FEBRERO
domingo	70437	156420
jueves	161108	345055
lunes	283169	385141
martes	275708	382720
miércoles	275497	402339
sábado	68344	148987
viernes	141935	304763

In [14]:
```
sorter = ['domingo', 'lunes', 'martes', 'miércoles', 'jueves', 'viernes', 'sábado']
sorterIndex = dict(zip(sorter,range(len(sorter))))
sorterIndex
```

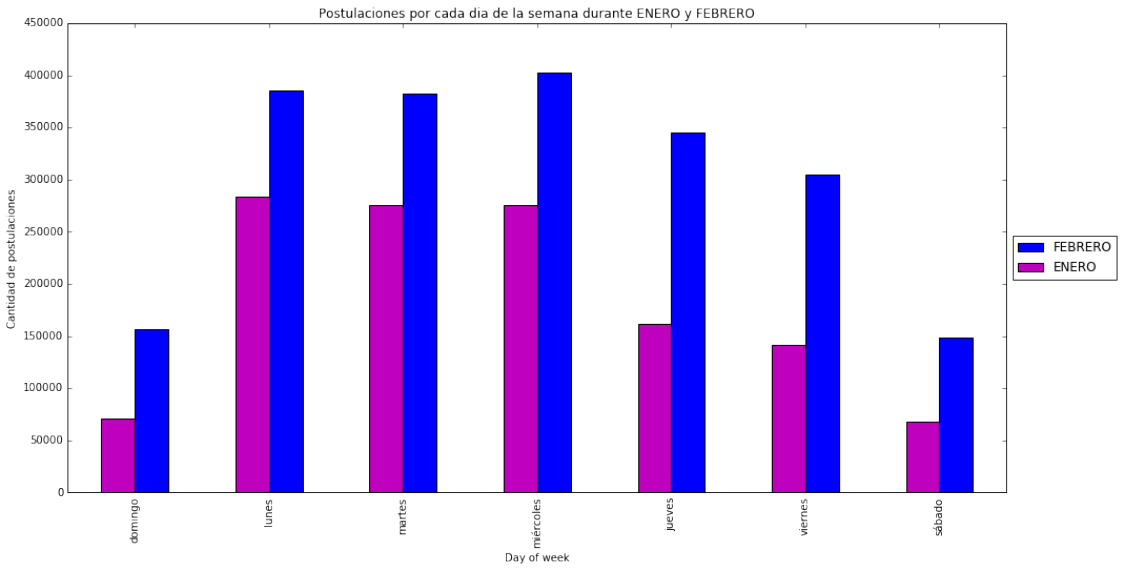
Out[14]:
```
{'domingo': 0,
 'jueves': 4,
 'lunes': 1,
 'martes': 2,
 'miércoles': 3,
 'sábado': 6,
 'viernes': 5}
```

In [15]:
```
selecc_postulaciones['Day_id'] = selecc_postulaciones.index
selecc_postulaciones['Day_id'] = selecc_postulaciones['Day_id'].map(sorterIndex)
selecc_postulaciones=selecc_postulaciones.sort_values(by=['Day_id'],ascending=True)
selecc_postulaciones
```

Out[15]:

Day of week	ENERO	FEBRERO	Day_id
domingo	70437	156420	0
lunes	283169	385141	1
martes	275708	382720	2
miércoles	275497	402339	3
jueves	161108	345055	4
viernes	141935	304763	5
sábado	68344	148987	6

In [16]:
```
#elimino la columna day_id
selecc_postulaciones=selecc_postulaciones[['ENERO','FEBRERO' ]]
f = plt.figure()
plt.title('Postulaciones por cada dia de la semana durante ENERO y FEBRERO', color='bla
ax = selecc_postulaciones.plot(kind='bar', stacked=False,figsize=(16,8),ax=f.gca(),colo
handles, labels = ax.get_legend_handles_labels()
plt.legend(handles[::-1], labels[::-1],loc='center left', bbox_to_anchor=(1.0, 0.5))
ax.set_ylabel('Cantidad de postulaciones');
ax.set_xlabel('Day of week');
plt.show()
```



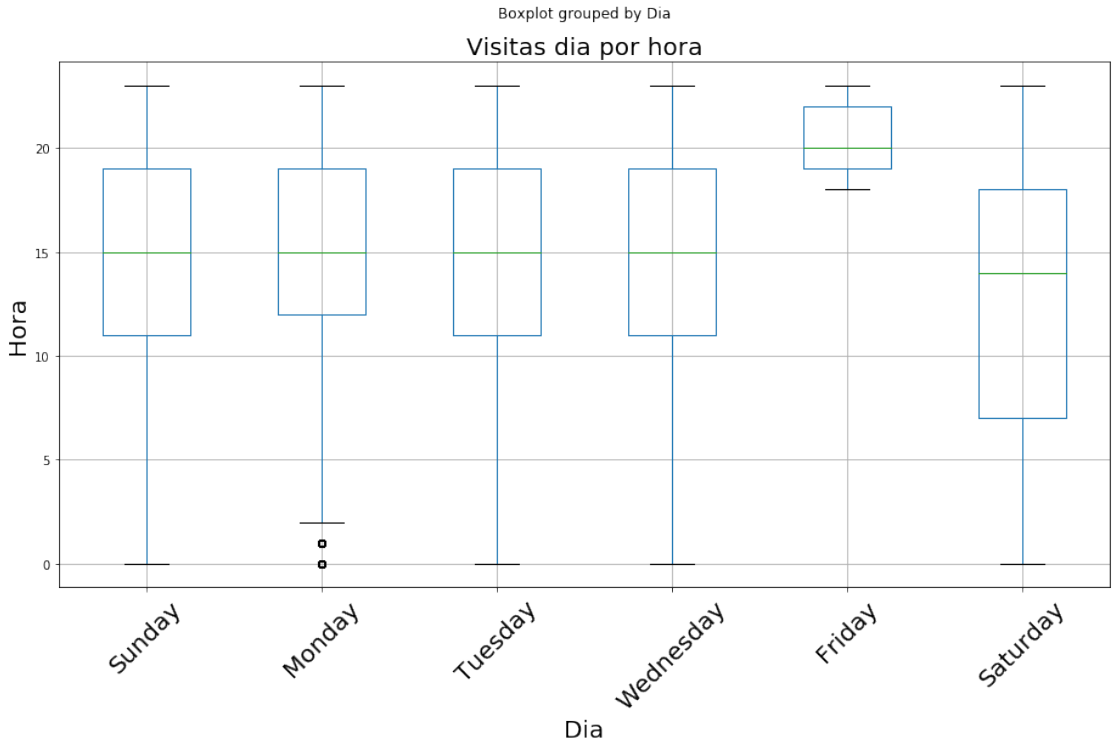
9 Conclusión:

9.1 Como observamos en el grafico tenemos que en el mes de Febrero para toda la semana se tuvo una mayor postulación a las avisos publicados.Ademas podemos ver que el dia miercoles para el mes de febrero fue un dia muy usado para la postulacion de los usuarios.

10 *8. grafico boxplot de visitas dia por hora*

10.1 Hipotesis: La hora con más actividad en internet se corresponde con la hora de comer, osea las 3PM

```
In [7]: visi3=vistasFebrero[['Dia','Hora']]
        ax =visi3.boxplot(rot=45, fontsize=20, figsize = (15,8), positions=[4,1, 5,0, 2, 3] ,by=

        ax.set_ylabel("Hora", fontsize=20)
        ax.set_title('Visitas dia por hora', fontsize = 20)
        ax.set_xlabel('Dia', fontsize = 20);
```



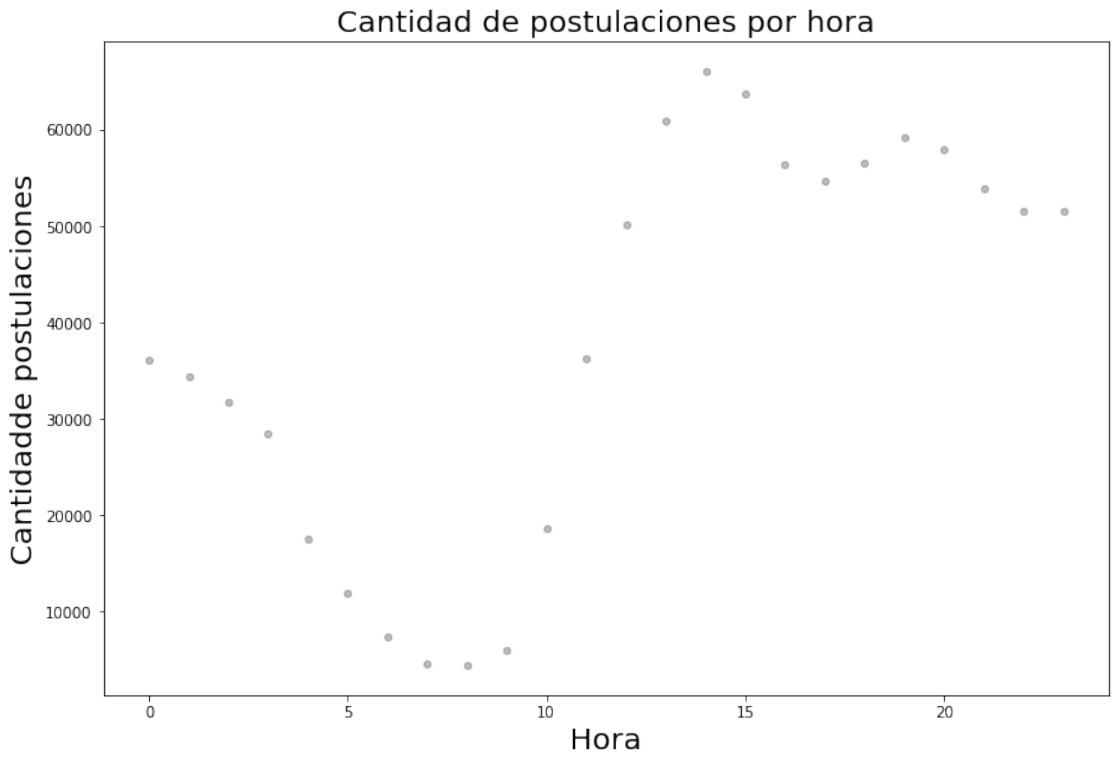
10.2 Conclusión:

10.2.1 Vemos que la mediana en horas los viernes es de 20 hrs, mientras que los otros dias es 15 hrs(excepto el sabado) vemos que en el sabado la parte baja de la caja es mayor que la de arriba; ello quiere decir que las horas comprendidas entre el 25% y el 50% de ese dia está más dispersa que entre el 50% y el 75%,idem con el viernes, y en los demas la mediana esta en el centro y es simetrica la distribucion de horas.Esto nos podria servir para detectar cuando es el momento de ofrecer anuncios importantes a los postulantes.

11 9. Hora en el dia que mas se postulan

11.1 Hipotesis: La hora con más actividad en internet se corresponde con la hora de comer, osea las 3PM,entonces a esa hora es mas probable que se postulen

In [18]:
```
visualizacionVistasPorDia2=vistasFebrero
visualizacionvistasporhora=visualizacionVistasPorDia2.groupby('Hora')['idpostulante'].a

ax =visualizacionvistasporhora.plot.scatter('Hora','count',alpha=0.25,figsize=(12,8),co


ax.set_ylabel("Cantidadde postulaciones", fontsize=20)
ax.set_title('Cantidad de postulaciones por hora', fontsize = 20)
ax.set_xlabel('Hora', fontsize = 20);
```



11.2 Conclusión:

11.3 La hipotesis fue correcta, alcanza un pico en las 15 horas, mientras que entre las 5 y 10 horas son las que menos se postulan,entonces podriamos mostrar anuncios que mas nos interesen mostrar a los postulantes durante esa hora.

12 *10. Area con mayor cantidad de publicaciones*

A partir de los dataFrames avisosCABA_GBA, postulantes y postulaciones obtengo un unico dataFrame con el cual vamos a trabajar de ahora en adelante

Primero mergeamos los dataFrames postulaciones y postulantes por el campo "idpostulante"

```
In [47]: data = pd.merge(postulantes, postulaciones, on='idpostulante', how='outer')
         data.head()
```

```
Out[47]:   idpostulante    nombre    estado fechanacimiento  sexo     idaviso  \
         0         NdJl  Posgrado  En Curso      1969-05-09  MASC  1112261212
         1         NdJl  Posgrado  En Curso      1969-05-09  MASC  1112273308
         2         NdJl  Posgrado  En Curso      1969-05-09  MASC  1112281548
         3         NdJl  Posgrado  En Curso      1969-05-09  MASC  1112260409
         4         NdJl  Posgrado  En Curso      1969-05-09  MASC  1112293547

              fechapostulacion
         0  2018-01-16 08:50:30
         1  2018-01-23 07:53:47
         2  2018-01-24 05:10:12
         3  2018-01-24 05:17:29
         4  2018-01-29 15:41:43
```

```
In [48]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5630562 entries, 0 to 5630561
Data columns (total 7 columns):
idpostulante        object
nombre              object
estado              object
fechanacimiento     object
sexo                object
idaviso             int64
fechapostulacion    object
dtypes: int64(1), object(6)
memory usage: 343.7+ MB
```

Y ahora mergeamos el dataFrame anterior con avisos por el campo "idaviso"s

```
In [49]: dataFrame = pd.merge(data, avisos, on='idaviso', how='outer')
         dataFrame.head()
```

Out[49]:

	idpostulante	nombre	estado	fechanacimiento	sexo	idaviso	\
0	NdJl	Posgrado	En Curso	1969-05-09	MASC	1112261212	
1	NdJl	Universitario	Graduado	1969-05-09	MASC	1112261212	
2	EppJmb	Universitario	En Curso	1960-06-20	MASC	1112261212	
3	EppJmb	Universitario	Graduado	1960-06-20	MASC	1112261212	
4	8Zr1GD	Terciario/Técnico	En Curso	1975-06-29	MASC	1112261212	

	fechapostulacion	idpais	\
0	2018-01-16 08:50:30	1.0	
1	2018-01-16 08:50:30	1.0	
2	2018-01-16 18:51:39	1.0	
3	2018-01-16 18:51:39	1.0	
4	2018-01-17 10:08:22	1.0	

	titulo	\
0	Jóvenes Profesionales - Responsable Administra...	
1	Jóvenes Profesionales - Responsable Administra...	
2	Jóvenes Profesionales - Responsable Administra...	
3	Jóvenes Profesionales - Responsable Administra...	
4	Jóvenes Profesionales - Responsable Administra...	

	descripcion	nombre_zona	\
0	<p>Nos encontramos en la búsqueda de un<strong...	Gran Buenos Aires	
1	<p>Nos encontramos en la búsqueda de un<strong...	Gran Buenos Aires	
2	<p>Nos encontramos en la búsqueda de un<strong...	Gran Buenos Aires	
3	<p>Nos encontramos en la búsqueda de un<strong...	Gran Buenos Aires	
4	<p>Nos encontramos en la búsqueda de un<strong...	Gran Buenos Aires	

	ciudad	mapacalle	tipo_de_trabajo	nivel_laboral	\
0	NaN	NaN	Full-time	Jefe / Supervisor / Responsable	
1	NaN	NaN	Full-time	Jefe / Supervisor / Responsable	
2	NaN	NaN	Full-time	Jefe / Supervisor / Responsable	
3	NaN	NaN	Full-time	Jefe / Supervisor / Responsable	
4	NaN	NaN	Full-time	Jefe / Supervisor / Responsable	

	nombre_area	denominacion_empresa
0	Administración	EDUCADO EN ARGENTINA
1	Administración	EDUCADO EN ARGENTINA
2	Administración	EDUCADO EN ARGENTINA
3	Administración	EDUCADO EN ARGENTINA
4	Administración	EDUCADO EN ARGENTINA

A partir de ahora vamos a trabajar con este dataFrame, el cual contiene la informacion de los anteriores

12.0.1 Verificamos sus tipos

In [50]:
```
dataFrame.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5632866 entries, 0 to 5632865
Data columns (total 17 columns):
idpostulante            object
nombre                  object
estado                  object
fechanacimiento         object
sexo                    object
idaviso                 int64
fechapostulacion        object
idpais                  float64
titulo                  object
descripcion             object
nombre_zona             object
ciudad                  object
mapacalle               object
tipo_de_trabajo         object
nivel_laboral           object
nombre_area             object
denominacion_empresa    object
dtypes: float64(1), int64(1), object(15)
memory usage: 773.6+ MB
```

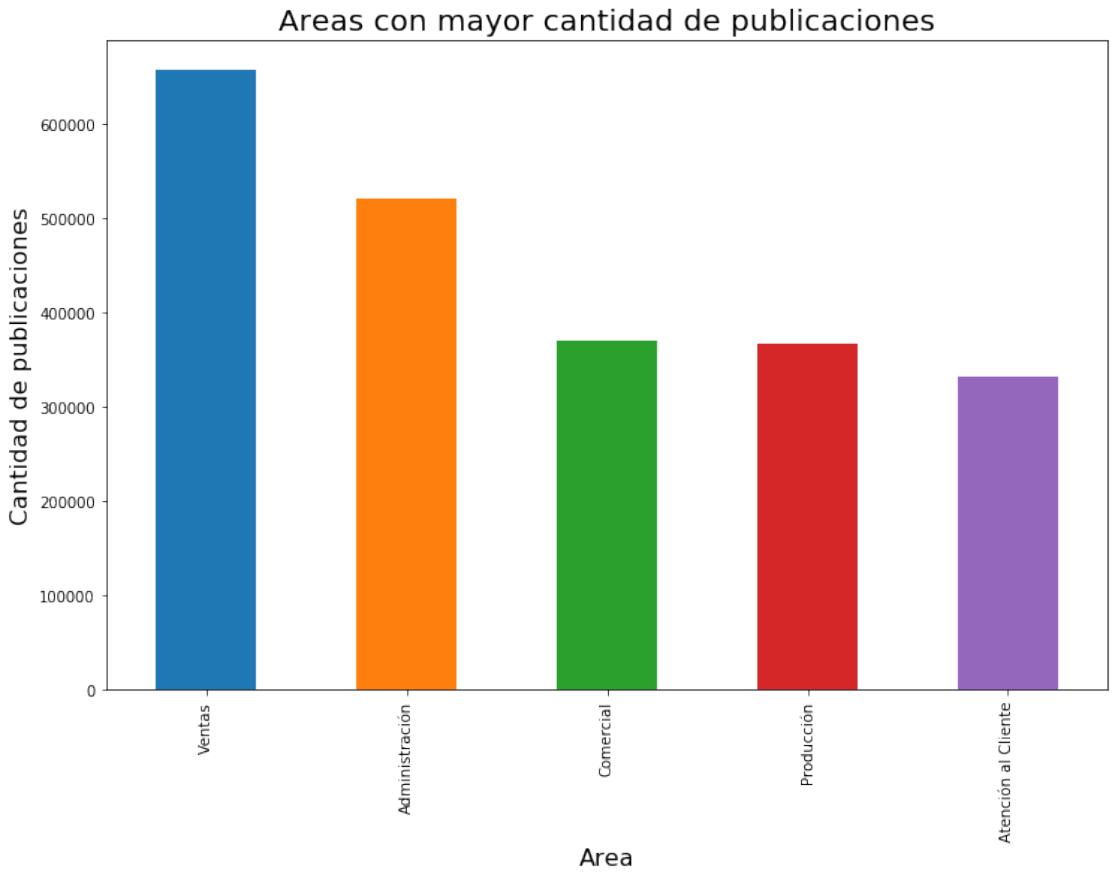
12.0.2 Agrupamos todos los avisos que corresponden al mismo area y vemos cuales son las que cuentan con mayor cantidad de publicaciones.

```
In [51]: dataFrame.groupby('nombre_area')['idaviso'].count().sort_values(ascending = False).head
```

```
Out[51]: nombre_area
         Ventas                 656473
         Administración         520098
         Comercial              370672
         Producción             367181
         Atención al Cliente    331374
         Name: idaviso, dtype: int64
```

Visualizamos los resultados con un grafico de barras

```
In [52]: dataFrame.groupby('nombre_area')['idaviso'].agg('count').sort_values(ascending = False)

         plt.title('Areas con mayor cantidad de publicaciones', fontsize=20);
         plt.xlabel('Area', fontsize=16);
         plt.ylabel('Cantidad de publicaciones', fontsize=16);
```



12.0.3 Nos interesaria ver la cantidad de publicaciones que tenemos segun la disponibilidad horaria de los postulantes

Para ellos agrupamos por la columna “tipo_de_trabajo”, y luego contamos la cantidad de avisos que hay de cada tipo (Full-time, Part-time y otros)

```
In [53]: dataFrame.groupby('tipo_de_trabajo')['idaviso'].count().sort_values(ascending  = False)
```

```
Out[53]: tipo_de_trabajo
         Full-time     4363571
         Part-time      662800
         Pasantia        27988
         Por Horas       17446
         Temporario      15205
         Name: idaviso, dtype: int64
```

Eliminamos las que tienen el campo “tipo_de_trabajo” nulo

```
In [ ]: disponibilidades_horarias = dataFrame
```

```
disponibilidades_horarias.dropna(subset=['tipo_de_trabajo'],inplace=True)

full_time = disponibilidades_horarias.loc[disponibilidades_horarias.tipo_de_trabajo.str.

part_time = disponibilidades_horarias.loc[disponibilidades_horarias.tipo_de_trabajo.str.

otros = disponibilidades_horarias.loc[~(disponibilidades_horarias.tipo_de_trabajo.str.co

sizes = [len(full_time), len(part_time), len(otros)]
nombres = ['Full-time', 'Part-time', 'Otros']

plt.figure(figsize=(6, 6))
plt.title('Porcentaje de publicaciones segun la disponibilidad horaria', fontsize=12)
plt.pie(sizes, labels=nombres, autopct='%1.1f%%', startangle=20, colors=['orange','green
plt.show()
```

13 *11. £Cuales son los avisos a las que mas personas se postulan?*

Agrupamos por el titulo del aviso, su id y la denominacion de empresa

```
In [ ]: dataFrame.groupby(['titulo','idaviso','denominacion_empresa'])['idpostulante'].count().s
```

Viendo los resultados, podriamos decir que hay avisos que estan duplicados. O sea que la empresa publico en una o mas ocasiones la misma busqueda laboral. Entonces podriamos agrupar simplemente todas los avisos que tengan el mismo titulo y que pertenezcan a la misma empresa (descartanto el idaviso)

```
In [ ]: dataFrame.groupby(['titulo','denominacion_empresa'])['idpostulante'].count().sort_values
```

Por lo cual, ahora los resultados cambian. Siendo la busqueda de Cajero/a - Part Time de Farmacity la que mas postulaciones tiene.

Visualizamos los resultados

```
In [ ]: dataFrame.groupby(['titulo','denominacion_empresa'])['idpostulante'].count().sort_values

        plt.title('Avisos con mayor cantidad de postulaciones', fontsize=16);
        plt.xlabel('Titulo/Empresa', fontsize=16);
        plt.ylabel('Cantidad de postulantes', fontsize=16);
```

Y ahora analizamos el aviso que esta en primera posicion

```
In [ ]: cajeroFarmacity = dataFrame
        cajeroFarmacity.dropna(subset=['titulo'],inplace=True)
        cajeroFarmacity = cajeroFarmacity.loc[cajeroFarmacity.titulo.str.contains('Cajero/a - Pa

        cajeroFarmacity['tipo_de_trabajo'].describe()
```

Aunque el titulo del anuncio diga que la busqueda es para Part-Time, aca se puede ver que en realidad se trata para disponibilidad Full-Time.

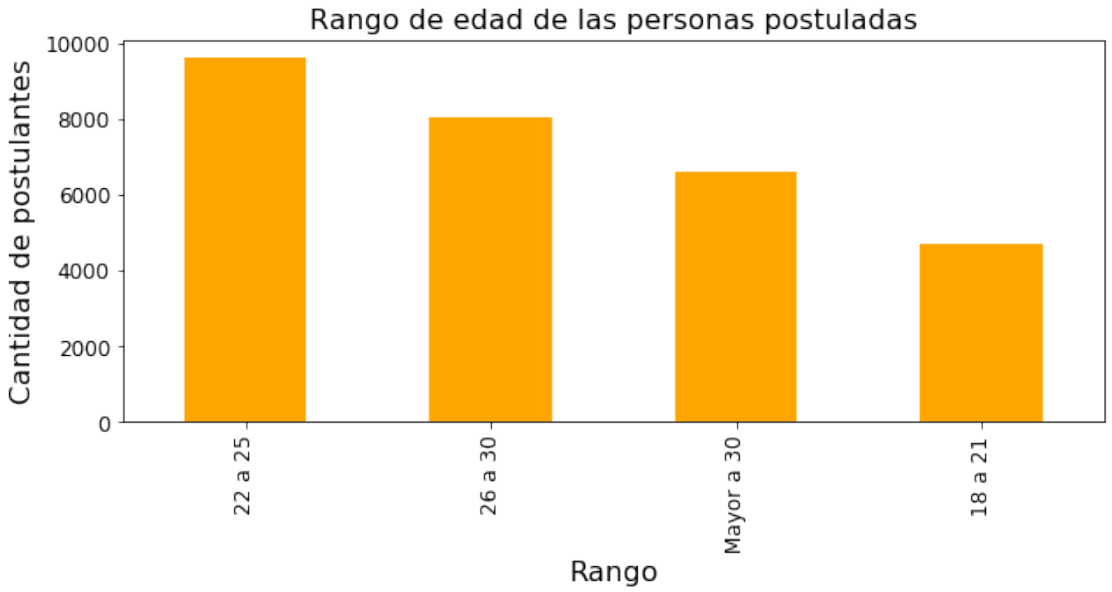
Nos centramos en el nivel educativo de los postulantes

```
In [ ]: cajeroFarmacity.groupby(['nombre','estado'])['idpostulante'].count().sort_values(ascendi
```

La gran mayoria se tratan de personas con estudios secundarios graduado. Seguido de otros con estudios universitarios en curso y mas atras graduados en otras cosas.

Seria interesante saber cual es el rango de edad de estas personas. Para ello utilizamos la calumna "fechanaciemto" de la cual optenemos el año de nacimiento de cada postulante

```
In [242]: ### #Convertimos la fecha fechanaciento en datetime y agregamos el campo año
          cajeroFarmacity['fechanacimiento'] = pd.to_datetime(cajeroFarmacity['fechanacimiento']
          cajeroFarmacity['anio'] = cajeroFarmacity['fechanacimiento'].map(lambda x: x.year)
          cajeroFarmacity.dropna(subset= ['anio'], inplace=True)
          cajeroFarmacity['anio'] = cajeroFarmacity['anio'].apply(lambda x: int(x))

          def rango(x):
              if (x[21]>2000):
                  return ("Menor a 18")
              elif (x[21]<=2000) & (x[21]>=1997):
                  return ("18 a 21")
              elif (x[21]<=1996) & (x[21]>=1993):
                  return ("22 a 25")
              elif (x[21]<=1992) & (x[21]>=1988):
                  return ("26 a 30")
              elif (x[21]<1988):
                  return ("Mayor a 30")

          cajeroFarmacity['rango_de_edad'] = cajeroFarmacity.apply(rango,axis=1)

          cajeroFarmacity.groupby('rango_de_edad')['idpostulante'].count().sort_values(ascending

          plt.title('Rango de edad de las personas postuladas', fontsize=16);
          plt.xlabel('Rango', fontsize=16);
          plt.ylabel('Cantidad de postulantes', fontsize=16);
```



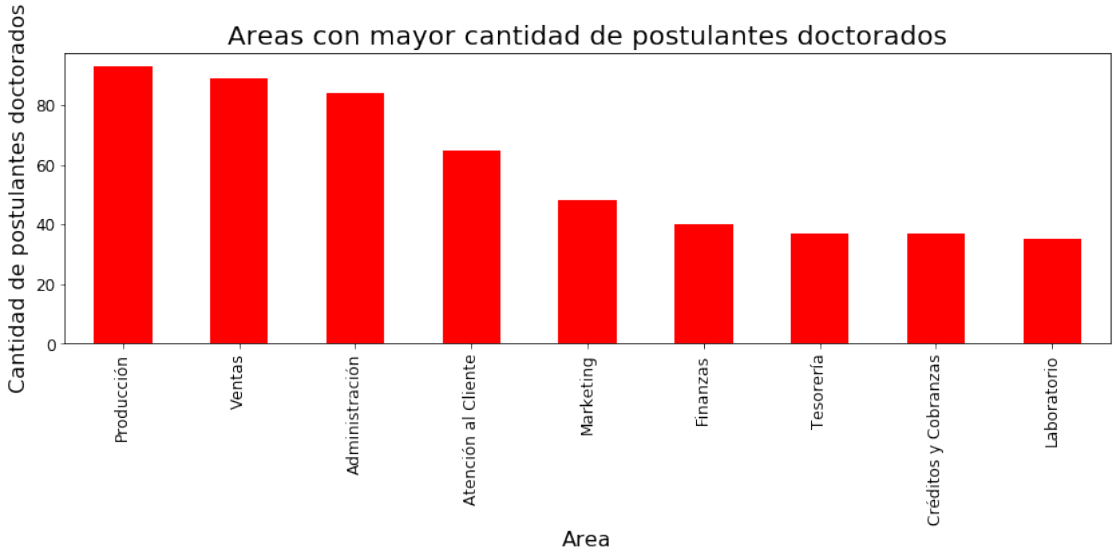
14 *12. Area con postulantes con nivel educativo mas alto*

Primero vemos cuales son los niveles educativos que existen

```
In [187]: dataFrame.groupby('nombre')['idpostulante'].count().sort_values(ascending = False).hea

Out[187]: nombre
          Secundario           1854451
          Universitario        1747126
          Terciario/Técnico     786795
          Otro                  460616
          Posgrado              106742
          Master                 53185
          Doctorado               3257
          Name: idpostulante, dtype: int64
```

Vemos que encabeza la lista los postulantes con nivel secundario (Ya sea en curso o terminado). Y luego de la lista podemos decir que el doctorado es el grado mas alto que se puede tener. Entonces vamos enfocarnos en este, para ver a cuales areas son las que mas se postulan

```
In [188]: doctorado = dataFrame.loc[(dataFrame.nombre.str.contains('Doctorado') & dataFrame.esta

          doctorado.groupby('nombre_area')['idpostulante'].count().sort_values(ascending = False

          plt.title('Areas con mayor cantidad de postulantes doctorados', fontsize=20);
          plt.xlabel('Area', fontsize=16);
          plt.ylabel('Cantidad de postulantes doctorados', fontsize=16);
```



---

15 *13. £Cual es el Sexo con mayor cantidad de postulaciones?*

15.0.1 Vale que el postulante se repita osea se postule a muchos avisos (En los meses de Enero y Febrero de 2018)

Hipotesis: 1 Trabajo. Las mujeres siguen sufriendo la discriminación laboral en cuanto a salarios y promoción. A igual trabajo, aún hay diferencias de sueldo entre mujeres y hombres, entonces al buscar una mejor opcion es problable que se postulen mas ,buscando mayor beneficio.

```
In [48]: postulacionesa = postulaciones
```

```
In [49]: IDs_duplicados=sum(postulacionesa.idpostulante.value_counts()>1)
         IDs_duplicados
```

```
Out[49]: 180941
```

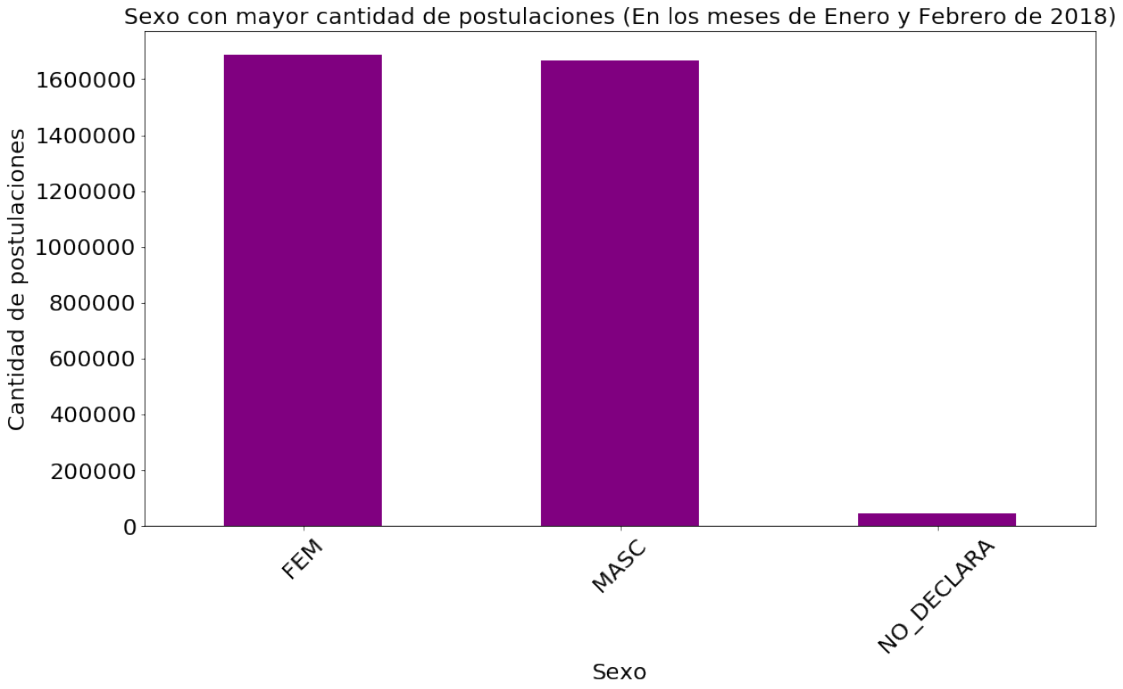
Vemos que hay duplicados en postulaciones

```
In [50]: postulantes_sexo=pd.merge(postulaciones,postulantesGeneroYEdad,on='idpostulante' , how=
```

```
In [51]: postulantes_sexo2=postulantes_sexo
```

```
In [52]: postulantes_sexo=postulantes_sexo.groupby('sexo')['idpostulante'].agg(['count'])
```

```
In [53]: graico =postulantes_sexo.plot(kind = "bar",\
                                       color = "Purple" ,\
                                       fontsize=20,figsize = (15,8), rot = 45, legend = False)
         graico.set_title('Sexo con mayor cantidad de postulaciones (En los meses de Enero y Feb
         graico.set_ylabel("Cantidad de postulaciones", fontsize=20)
         graico.set_xlabel("Sexo", fontsize = 20)

Out[53]: <matplotlib.text.Text at 0x7fbc46797b10>
```



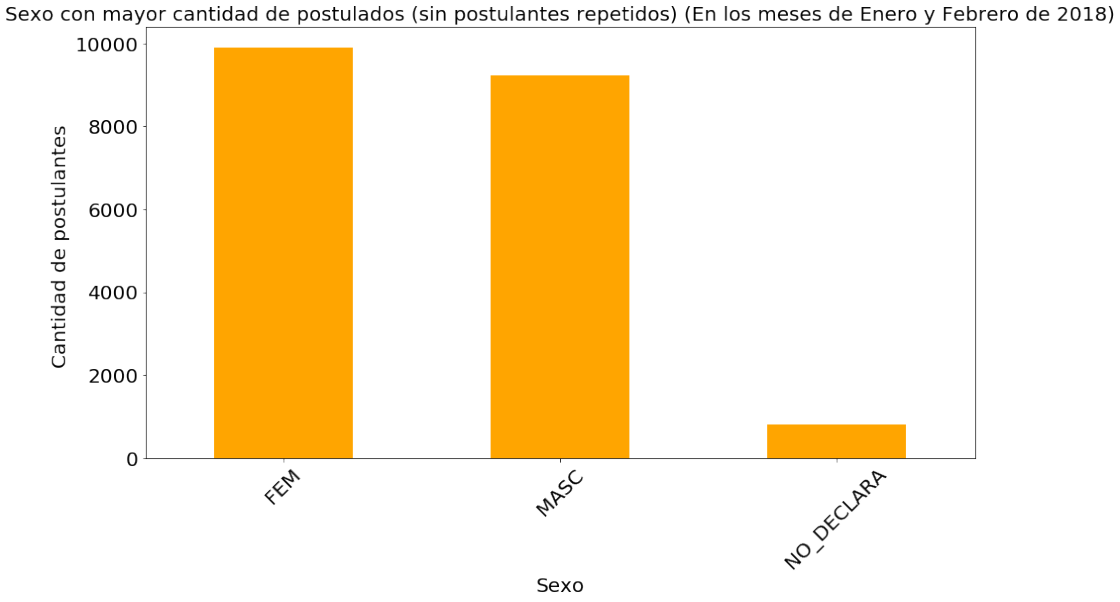
15.1 ** CONCLUSION:Por lo cual se puede ver que la mujeres se postulan en casi igual cantidad que los hombres,por lo tanto apriori viendo este grafico,habra igual flujo de postulaciones en los avisos de parte del los dos generos. **

16 *14. Sexo con mayor cantidad de postulados (sin repeticion de postulante)*

16.0.1 Es decir que solo se muestra una postulacion a un aviso (En los meses de Enero y Febrero de 2018)

```
In [54]: postulantes_sexo_sinrepeticiones=postulantes_sexo2.drop_duplicates(subset=['idpostulant

In [55]: IDs_duplicados=sum(postulantes_sexo_sinrepeticiones.idpostulante.value_counts()>1)
         IDs_duplicados

Out[55]: 0
```

```
In [57]: postulantes_sexo_sinrepeticiones=postulantes_sexo_sinrepeticiones.groupby('sexo')['idpo

In [58]: grafico=postulantes_sexo_sinrepeticiones.plot(kind = "bar",\
                                    color = "Orange" ,\
                                    fontsize=20,figsize = (15,8), rot = 45, legend = False)

         grafico.set_title("Sexo con mayor cantidad de postulados (sin postulantes repetidos) (E
         grafico.set_ylabel("Cantidad de postulantes ", fontsize=20)
         grafico.set_xlabel("Sexo", fontsize = 20)

Out[58]: <matplotlib.text.Text at 0x7fbc46784f10>
```

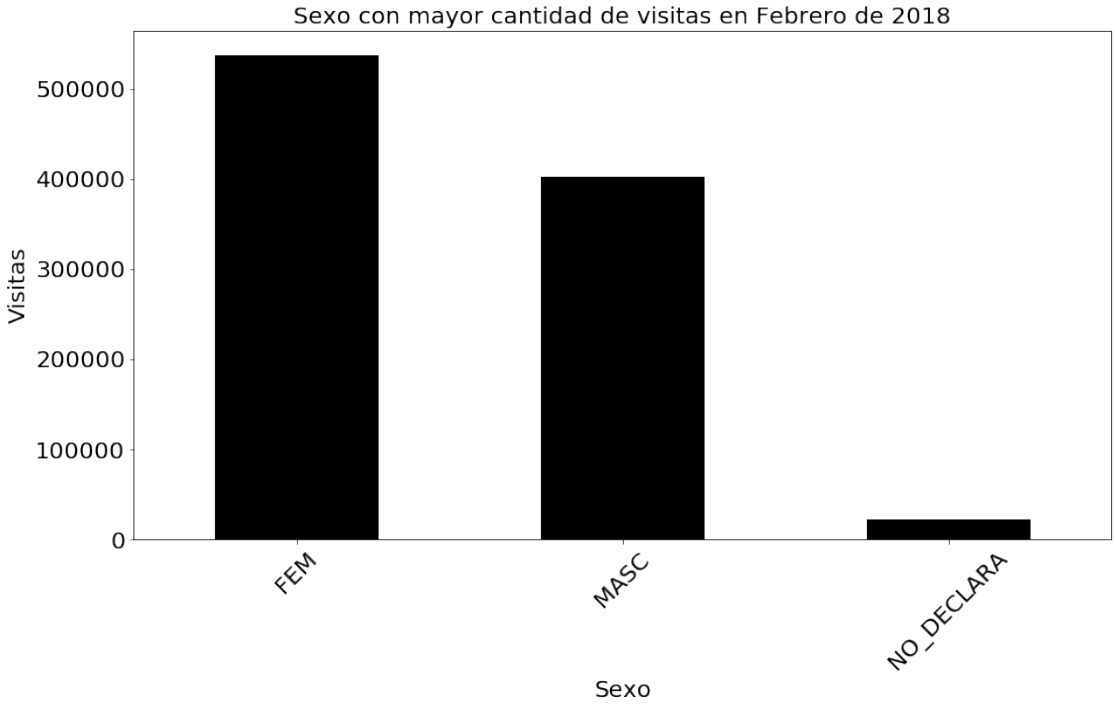


16.1 ** CONCLUSION: hay mayor postulacion de las mujeres en enero y febrero,(recordando que este grafico solo muestra que un postulante se postula a un aviso) .esto dice junto con el grafico de postulaciones con repeticiones ,que los hombres se postulan a muchos avisos,mientras que las mujeres en menor medida. **

17 *15. Sexo con mayor cantidad de visitas (Febrero de 2018)*

```
In [28]: sexo_visitas=pd.merge(vistas,postulantesGeneroYEdad,on='idpostulante' , how='inner')

In [29]: sexo_sitas=sexo_visitas.groupby('sexo')['idpostulante'].agg(['count']).sort_values(by='
                                    color = "Black" ,\
                                    fontsize=20,figsize = (15,8), rot = 45, legend = False)

         sexo_sitas.set_title("Sexo con mayor cantidad de visitas en Febrero de 2018",fontsize=2
         sexo_sitas.set_ylabel("Visitas", fontsize=20)
         sexo_sitas.set_xlabel('Sexo', fontsize = 20)
```

```
Out[29]: <matplotlib.text.Text at 0x7fbc47b03c50>
```



17.1 ** Conclusion:nuestra hipotesis inicial fue erronea, fueron las mujeres las que mas visitaron los avisos,entonces la probabilidad que un aviso sea visto por una mujer es mayor. **

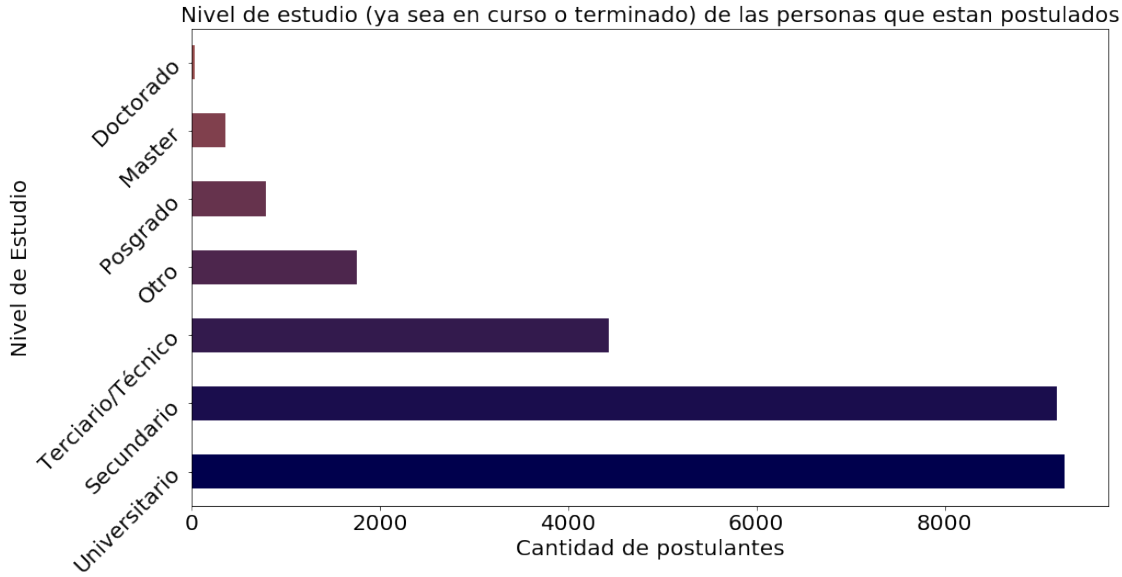
18 *16. Nivel de estudio (ya sea en curso o terminado) de las personas que estan postulados*

18.0.1 Hipotesis: A Mayor grado educativo , mas facil es conseguir empleo,por lo tanto sera menor la cantidad de personas de alto nivel educativo que se registren a buscar trabajo en los avisos.

```
In [60]: educacion2=postulantesEducacion

In [61]: IDs_duplicados=sum(educacion2.idpostulante.value_counts()>1)
         IDs_duplicados

Out[61]: 80276

In [62]: educacion2.estado.value_counts()

Out[62]: Graduado      194474
         En Curso       78531
         Abandonado     25226
         Name: estado, dtype: int64
```

significa que el postulantes podrian haber dejado la carrera, cambiado o comenzado una nueva, Por eso hay duplicados

```
In [63]: postulaciones3=postulaciones

In [64]: postulantessinrepeticiones=postulaciones3.drop_duplicates(subset=['idpostulante'], keep
         #Queremos los que se postulan, por eso eliminamos repetidos

In [65]: postulantes_nivelestudio=pd.merge( postulantessinrepeticiones,postulantesEducacion,on='

In [66]: postulantes_nivelestudio=postulantes_nivelestudio.groupby('nombre')['idpostulante'].agg
         my_colors = [(x/10.0, x/20.0, 0.30) for x in range(len(postulantes_nivelestudio))] # <-
         grafico =postulantes_nivelestudio.plot(kind = "barh",\
                                        color = my_colors ,\
                                        fontsize=20,figsize = (15,8), rot = 45, legend = False)

         grafico.set_title("Nivel de estudio (ya sea en curso o terminado) de las personas que e
         grafico.set_xlabel('Cantidad de postulantes', fontsize=20);
         grafico.set_ylabel("Nivel de Estudio", fontsize=20)

Out[66]: <matplotlib.text.Text at 0x7fbc47b7d250>
```



18.1 ** CONCLUSION: El nivel de estudio mas popular entre los postulados fueron el universitario,por lo tanto la hipotesis no fue tan correcta,porque los universitarios salieron con mas postulaciones.Por lo tanto es mas probable que un secundario o universitario se postule en algun aviso. **

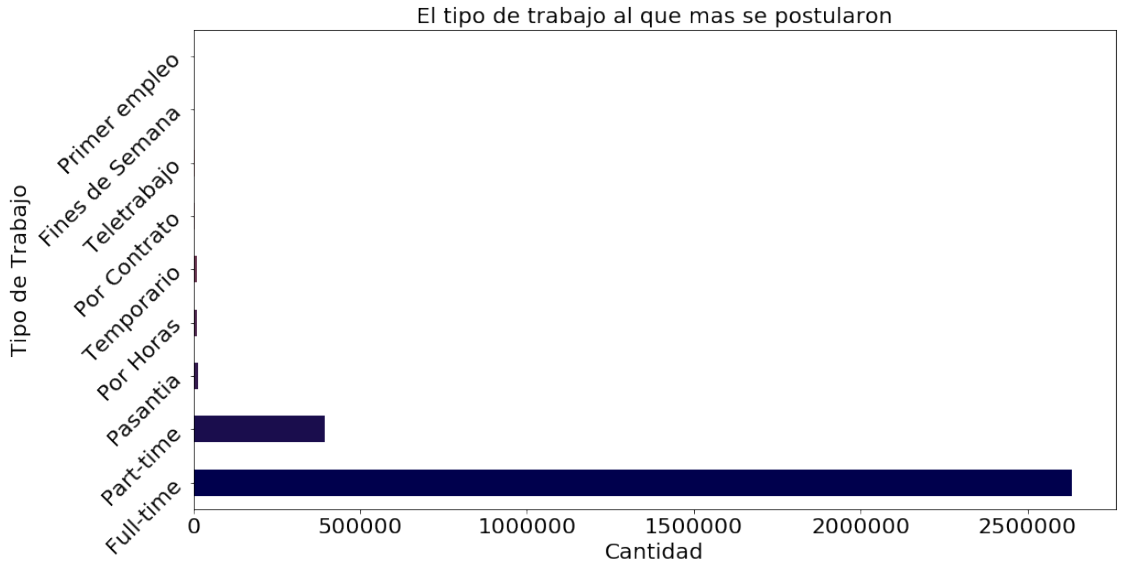
19 *17. El tipo de trabajo al que mas se postularon*

19.0.1 Hipotesis:Cuando se trabaja en relación de dependencia, las modalidades más comunes son las del trabajo de tiempo completo (full-time) y el trabajo de medio tiempo (part-time),veremos si nuestra hipotesis es correcta.

In [19]:
```
postulantes_trabajo=pd.merge(postulaciones,avisosDetalle,on='idaviso' , how='inner')
```

In [20]:
```
postulantes_trabajo=postulantes_trabajo.groupby('tipo_de_trabajo')['idpostulante'].agg(
my_colors = [(x/10.0, x/20.0, 0.30) for x in range(len(postulantes_trabajo))] # <-- Qui
grafico = postulantes_trabajo.plot(kind = "barh",\
                                   color = my_colors ,\
                                   fontsize=20,figsize = (15,8), rot = 45, legend = False)

grafico.set_title("El tipo de trabajo al que mas se postularon",fontsize=20)
grafico.set_xlabel('Cantidad', fontsize=20);
grafico.set_ylabel("Tipo de Trabajo", fontsize=20)
```

Out[20]:
```
<matplotlib.text.Text at 0x7fbc475d3590>
```



19.1 ** Conclusion: los tipos de trabajo mas postulados fueron el full time y part-time, entonces los postulantes van a ser proclives a postularse en estos tipos de trabajo partime y fulltime **